Spring 3-15-2010

Candidate Tracking System

Dave Bishop
Dakota State University

CANDIDATE TRACKING SYSTEM

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

Spring 2010

By

Dave Bishop

Project Committee:

Dr. Wayne Pauli

Dr. Ronghua Shan

Denise Luckhurst

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Dave Bishop

Master's Project Title: Candidate Tracking System

Faculty supervisor: Wayne E Pauls Date: March 10, 2010

Committee member: Ronghua Shan Date: 3/12/2010

Committee member: Denise Lockhurst Date: 3/16/2010

ACKNOWLEDGMENT

I would like to acknowledge the following individuals for assisting with the completion of this project:

Dr. Wayne Pauli as my project supervisor for his guidance and support for my project;

Dr. Ronghua Shan for being a part of my project committee;

Denise Luckhurst for being a part of my project committee;

Norm Harrod for providing assistance getting some of the enterprise house-keeping items setup;

Scott Darnall for the work he did on his data access code generator, it literally saved me writing thousands of lines of code;

Lastly, I would also like to acknowledge my wife Barb, and children Nathan and his wife Sara, Zak, Cody and Lindsey for supporting me during the time spent working toward my MSIS degree.

ABSTRACT

This project automates the candidate tracking process used by the Development Division of the Bureau of Information and Telecommunications (BIT) for the State of South Dakota. The Development Division employs approximately 120 to 130 staff. Due to the size of the Division it is common to have one or more positions open at any point in time. Consideration is given to supporting other BIT Divisions' use of the resulting system as well as other agencies.

The Division uses a manual process to track candidates for these open positions. The manual process is cumbersome, inefficient and time consuming requiring the updating of a shared spreadsheet and numerous emails between Development Team Managers, the Division Director and the Bureau of Personnel Representative.

The Division requested to automate this process in order to attract and hire the most qualified candidates by providing timely feedback to applicants and reducing the time from application to offer for qualified candidates while maintaining accurate information about the recruiting process.

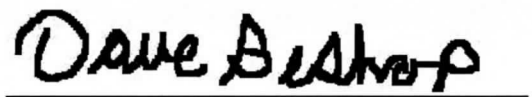
This project used the Scrum methodology and was executed in multiple sprints. The Scrum philosophy proved valuable while making scheduling decisions, feature trade-offs and maintaining focus on the business goals.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,



Dave Bishop

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Statement of the problem

The Development Division currently has a manual process in place to manage candidates. The process is somewhat complex and requires many steps to process a candidate. In addition there are many communications between the Bureau of Personnel, the management team and the candidates. The complexity of this manual process introduces a significant amount of time which reduces the ability to reach the best candidates in a timely manner. Occasionally candidates stall out in the process due to missed communication or dropped tasks further slowing the processing of the candidates.

In addition to the need to present timely offers to qualified candidates the Division must manage the risk associated with selective hiring and consequently must maintain historical information about the candidate selection process in case it is needed in court to defend its hiring practices. Applicant information, interview dates and resulting notes along with status notifications sent to candidates must be kept in order to manage risk.

Therefore, the Development Division must find a way to reduce the time-to-offer for well qualified candidates while maintaining enough historical data to mitigate potential hiring litigation.

Objectives of the project

The goal of this effort is to create a database driven, web-based, Candidate Tracking system for the State of South Dakota's Bureau of Information and Telecommunications that will reduce the time-to-offer to qualified candidates while protecting the organization from potential litigation. The sponsor of the project recognizes that other divisions and other agencies within the Executive Branch may experience similar needs. With that in mind, where possible, the design should accommodate a wider audience of other divisions and agencies.

Approach

This paper will reflect the approach taken on the project. The project was executed using the Scrum methodology consequently the work was carried out in sprints. A sprint is a self-contained mini-project which turns a set of Product Backlog Items into tested, working application functionality. Each sprint will be documented providing the sprint goals, a user class list, a use case list with actors, a use case context diagram and use case definitions.

Lessons learned will be reviewed. These include lessons about project management using the Scrum methodology, the value of a data access code generator and the difficulty of artifact maintenance. The appendix contains sample code including database stored procedures, middle tier VB.Net classes and user interface .aspx and .aspx.vb files. The appendix also contains some sample images of the final user interface.

Definition of Terms

The following table of terms and definitions are not intended to be a rigorous academic definition; rather they are working terms defined as used within this paper.

Table 1 - Terms and Definitions

Term	Definition
Scrum methodology	An iterative approach to delivering software organized into time limited phases with the goal of completing working software at the end of each iteration.
Sprint	A software development iteration that delivers the highest value features within a fixed time period, usually between two to four weeks in duration.
Use Case	A technology agnostic description of how a user performs a process to accomplish a business goal.
Trigger	A situation within a business that causes a user to exercise a use case to achieve their goal.
Actor	A generic term similar to a role which represents a one or more users of a process who perform a use case to achieve their goal.
Artifact	Documentation outside of the software code. Typically Word documents that contain requirements, use cases, design information, data dictionaries and testing scenarios.
Normal Flow	This is the primary path through a use case which allows the user to accomplish the goal for the use case.
Alternative Flow	This is a deviation from the primary path caused by unusual circumstances but important to understanding how the process is completed for varying scenarios.
Context Diagram	A figure that depicts how the actors and system interact via use cases.
Time Boxing	A project management technique that uses time as the limiting factor for determining the scope that is accomplished for a particular phase of the project. Rather than say features x, y and z will be implemented and work will continue until all of the selected features are completed, time boxing says we will finish in 3 weeks with how ever many features we can complete in that amount of time.

CHAPTER 2

SPRINT 0 – APPLICATION FOUNDATION

Sprint Goals

The goal of the initial sprint is somewhat different from all of the other sprints. The Application Foundation sprint is used to create the fundamental environment on which all other sprints depend. In this sprint error handling, exception handling, underlying user interface elements and the initial version of the data model are all designed and implemented as appropriate. Each of the following sprints will incorporate the deliverables of this sprint and to a large extend will provide the test cases to ensure these foundational elements are working as expected.

Requirements Information

User Class List

The User Class List enumerates and describes the categories of users that will interact with the application in this sprint.

Table 2 –Sprint 0 User Class List

User Class Name	User Class Description
User	In Sprint 0 User is an abstraction of all future system users. The use cases of Sprint 0 are foundational in nature.
System	The Candidate Tracking System

Use Case List with Actors

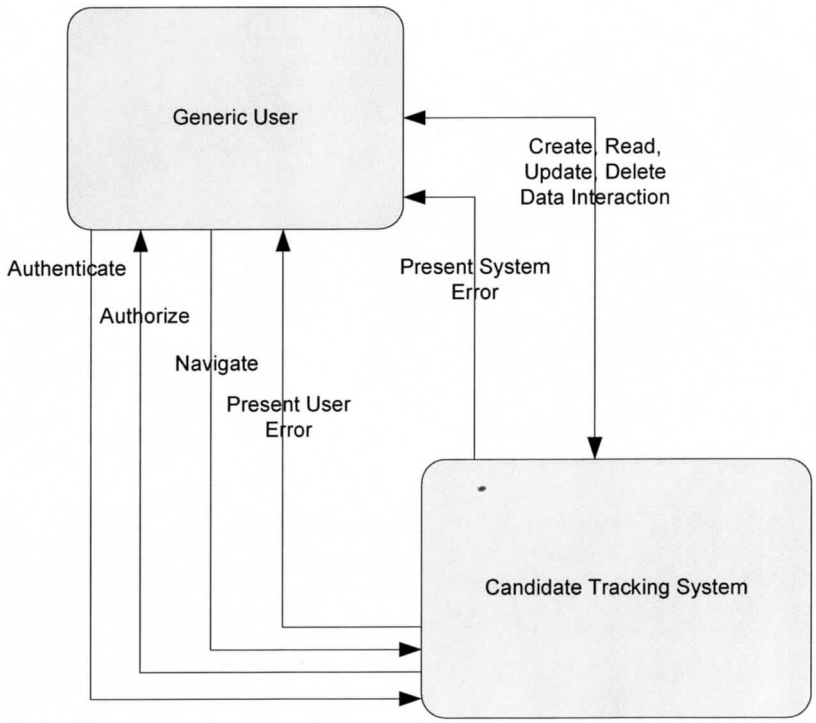
The Use Case List with Actors associates the User Classes with the Use Case Names providing a quick overview of the the users and functionality of the sprint.

Table 3 – Sprint 0 Use Case List

Use Case Id	Actors	Use Case Name
UC-0.1	User	Authenticate to system
UC-0.2	System	Authorize feature
UC-0.3	User	Navigate to feature
UC-0.4	System	Present user error
UC-0.5	System	Present system error
UC-0.6	User	Create, Read, Update, Delete data interaction

Use Case Context Diagram

Figure 1 – Sprint 0 Use Case Context Diagram



Use Case Definitions

Use Case ID:	UC-0.1		
Use Case Name:	Authenticate to system		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/1/2009	Date Last Updated:	11/1/2009
Actors:	User		
Description:	The system validates access for the user based on the users credentials		
Trigger:	Attempted user access to the system		
Preconditions:	The users has been set-up for system usage		
Postconditions:	The user is able to access the features for which they are authorized		
Normal Flow:	1. The user attempts to access the system 2. The system validates that the user has access to the system 3. The user is granted access to the system		
Alternative Flows:	3a. The user is not setup for access to the system 3a1. A message is presented to the user informing them they do not have access 3a2. The user is not granted access to any system features		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-0.2		
Use Case Name:	Authorize feature		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/1/2009	Date Last Updated:	11/1/2009
Actors:	System		
Description:	The system filters the available features and functionality based on the authenticated user's role.		
Trigger:	Successful access to the system by a user		
Preconditions:	1. The user has successfully authenticated 2. The user is assigned to a role		
Postconditions:	The user is able to access the features for which they are authorized		
Normal Flow:	1. The user successfully access the system 2. The system adjusts the available features based on the users role		
Alternative Flows:			
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-0.3		
Use Case Name:	Navigate to feature		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/1/2009	Date Last Updated:	11/1/2009
Actors:	User		
Description:	The user is able to select the desired feature		
Trigger:	Attempted user access to features		
Preconditions:	The user has successfully authenticated		
Postconditions:	The user is able to access the features for which they are authorized		
Normal Flow:	1. The system displays the authorized features 2. The user selects the desired feature to execute		
Alternative Flows:			
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-0.4		
Use Case Name:	Present User Error		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/1/2009	Date Last Updated:	11/1/2009
Actors:	System		
Description:	The system presents user error information and corrective actions to the user and allows the user to make changes to eliminate the error condition(s).		
Trigger:	The user enters incorrect data		
Preconditions:	1. The user has successfully authenticated to the system 2. The user is authorized to execute the selected feature 3. The user has entered incorrect information		
Postconditions:	1. The user can identify the incorrect information 2. The user can correct the identified information		
Normal Flow:	1. Incorrect information is detected by the system 2. One or more messages are presented to the user 3. The user is allowed to correct the data		
Alternative Flows:	3a. The user selects to cancel changes 3a1. No changes are made permenant		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-0.5		
Use Case Name:	Present System Error		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/1/2009	Date Last Updated:	11/1/2009
Actors:	System		
Description:	The system presents system error information to the user.		
Trigger:	The system detects an internal system error.		
Preconditions:			
Postconditions:	A message is presented with as diagnostic information about the system error		
Normal Flow:	1. The system detects an internal system error 2. The system displays diagnostic error information		
Alternative Flows:			
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	Low		
Business Rules:			

Use Case ID:	UC-0.6		
Use Case Name:	Create, Read, Update, Delete data interaction		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/1/2009	Date Last Updated:	11/1/2009
Actors:	User		
Description:	The user requests through system features to read or modify information maintained by the system and the system responds to the request.		
Trigger:	A request to create, read, update or delete information.		
Preconditions:	1. The user is authorized to execute the desired feature. 2. The requested operation does not violate data integrity.		
Postconditions:	1. Create – The data is inserted to the database 2. Read – The data is retrieved from the database 3. Update – The data is modified in the database 4. Delete – The data is removed from the database		
Normal Flow:	1. The user requests a data operation 2. The operation is performed against the database		
Alternative Flows:	2a. A connection to the database is unavailable 2a1. The Present System Error use case is executed 2b. The requested operation violates a data integrity constraint 2b1. The Present User Error use case is executed		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	Low		
Business Rules:			

Design Information

Architecture Description

Production Hardware and Software Environment

The Candidate Tracking System will run on the State of South Dakota's standard intranet infrastructure. The production web farm is composed of four virtual servers. The State currently uses Microsoft's Hyper-V virtualization software inside of Windows Server R2. Each virtual server runs an instance of Microsoft's Internet Information Server 6.0. The four IIS servers are load balanced behind an F5 Network BIG-IP network appliance.

The data storage will be handled by Microsoft's SQL Server 2005 relational database using HP SAN attached disk resources. The SQL Server runs on a pair of HP Proliant 460C blade server which are configured as a failover cluster using Microsoft's Clustering Services. This clustering provides high availability by automatically moving the services to the available node if one node fails. The two nodes and associated SAN storage are geographically distributed to prevent an outage from a site failure.

The primary development software used for the project is Visual Studio 2008 and the .Net Framework 3.5. Visual Studio provides the Integrated Development Environment for creating the application classes and user interface elements. Visual Studio 2008 integrates with Microsoft's Visual Source Safe 6.0 which will be used for source code versioning.

Authentication

The Candidate Tracking System (CTS) will utilize Internet Information System as the web server technology. It will also utilize Windows Authentication to identify the user requesting access to the application. Through Windows Authentication CTS will obtain the users identity and validate the user's access against its data to verify that the user should have access to the system.

Authorization

Once a user has authenticated, CTS will utilize role based authorization implemented using a Role database table and a PageRole class to compare the currently logged in user's role with the application's web pages and menu items to determine whether the user has access to the resource. Each user will have a default role, and may have additional roles to choose from. A user will only be able to operate in a single role at any point in time.

The PageRole class in conjunction with the user's active role will be used by the system to show, hide, enable and disable features and functionality throughout the application.

Navigation

The Candidate Tracking System will employ a treeview control and sitemap, both of which are native components within Microsoft Visual Studio 2008. The sitemap defines the menu items available and the URL's of the associated pages. The treeview reads the sitemap and populates its nodes accordingly.

The user manipulates the nodes on the treeview to reveal the features available based on their current role. The treeview will be located on the left side of the page to provide a consistent location and means of navigation throughout the application.

Data Interaction

A colleague, Scott Darnall, provided a utility application that he developed which interrogates a physical RDBMS and generates custom stored procedures and VB.Net classes for data access. The database must follow a few simple table and column naming conventions in order for the utility to work. CTS used this utility to generate the entire data access tier of the application.

The utility generates a set of programming items for each table in the database and uses index and foreign key information in the process. The foundation of the utility is based on a set of stored procedures for insert, update, delete and query. Next a class is generated called a data class with the prefix "CD" combined with the table name. This class handles calling the stored procedures and formatting the results into a .Net DataSet. This class has properties for each column in the database table. Another class is the business class. These classes are generated with a "CB" prefix followed by the table name. This class is the interface for the rest of the application. The application should never directly instantiate the

data classes, only the business classes. The final object generated by the utility is the collection class designated with a prefix of "CC" followed by the table name. This class is used to get sets of records, usually lists based on foreign keys or non-unique indexes.

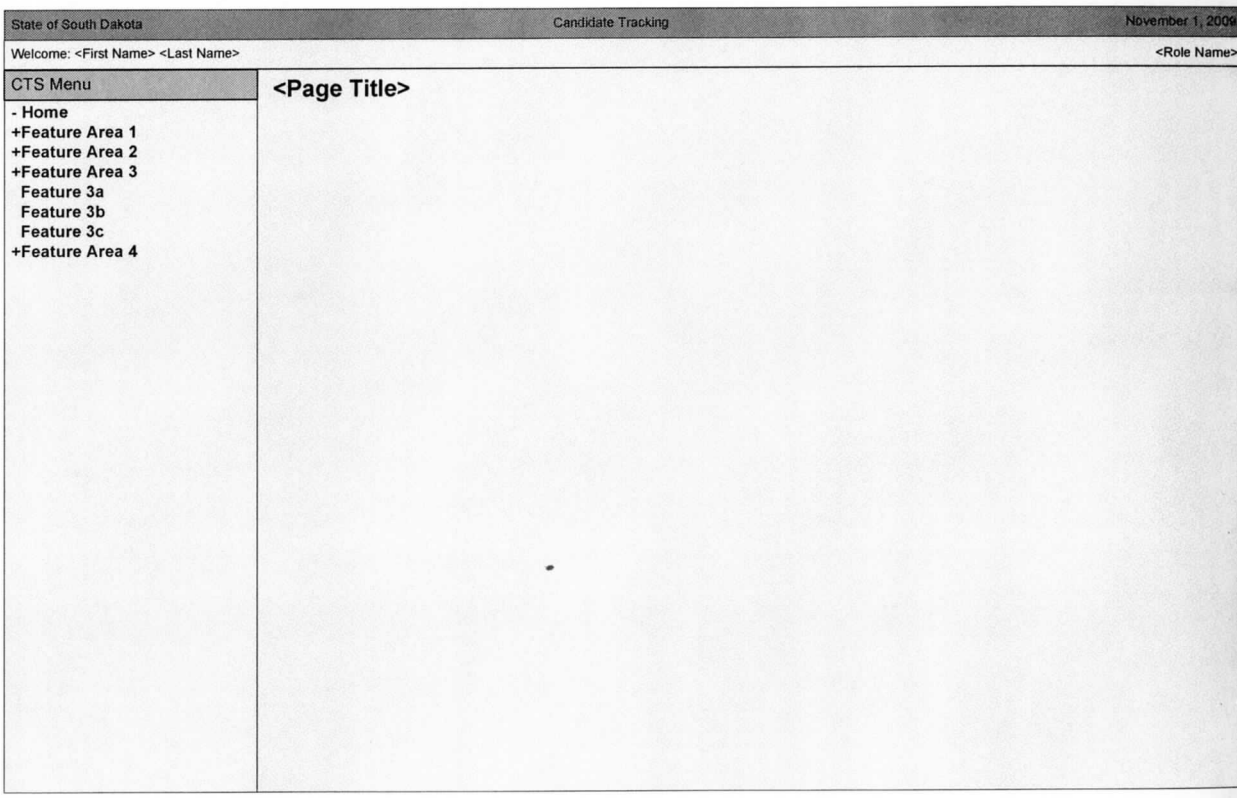
Present User Error

User error handling will be displayed by utilizing a div at the top of the page and populating it with the appropriate error text in a red font. Each page has a an error div for a consistent approach programmatically and from a user perspective for error handling.

Page Level Design and Behavior

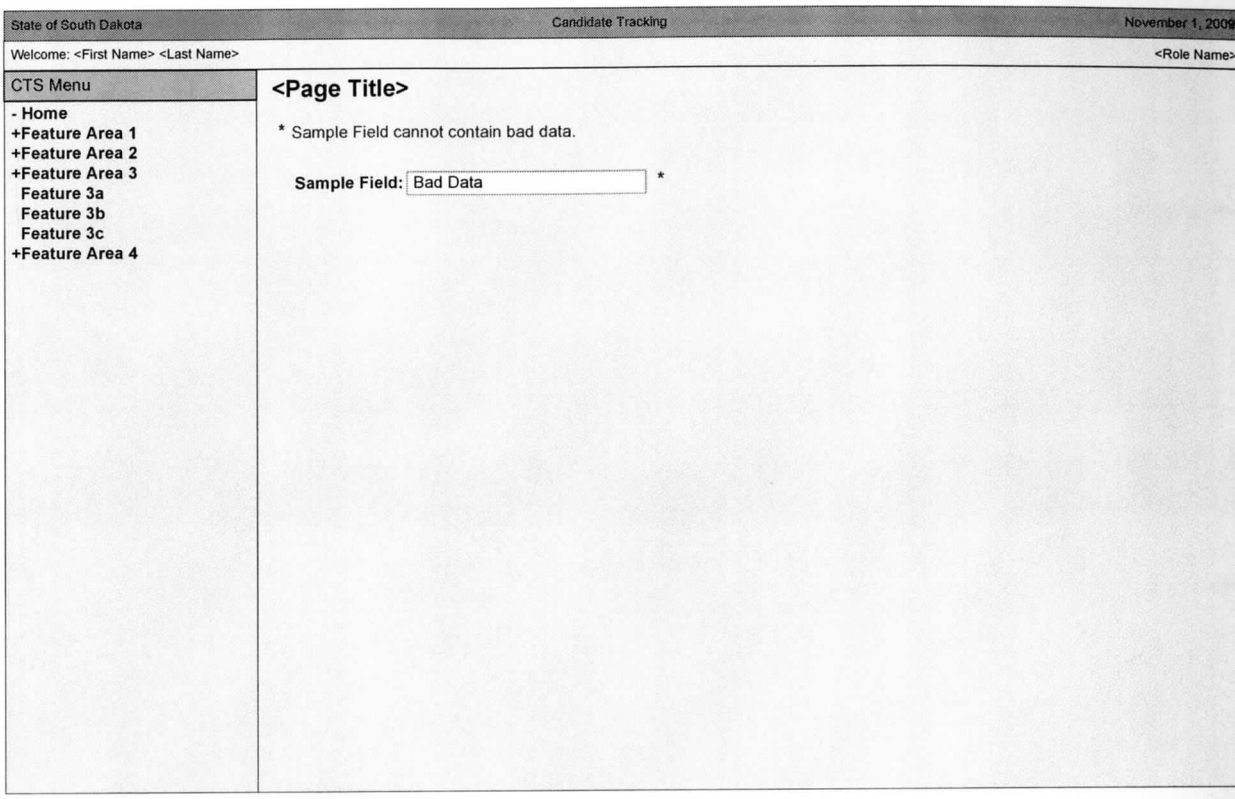
Treeview Navigation

Figure 2 - Treeview Navigation Example



User Error

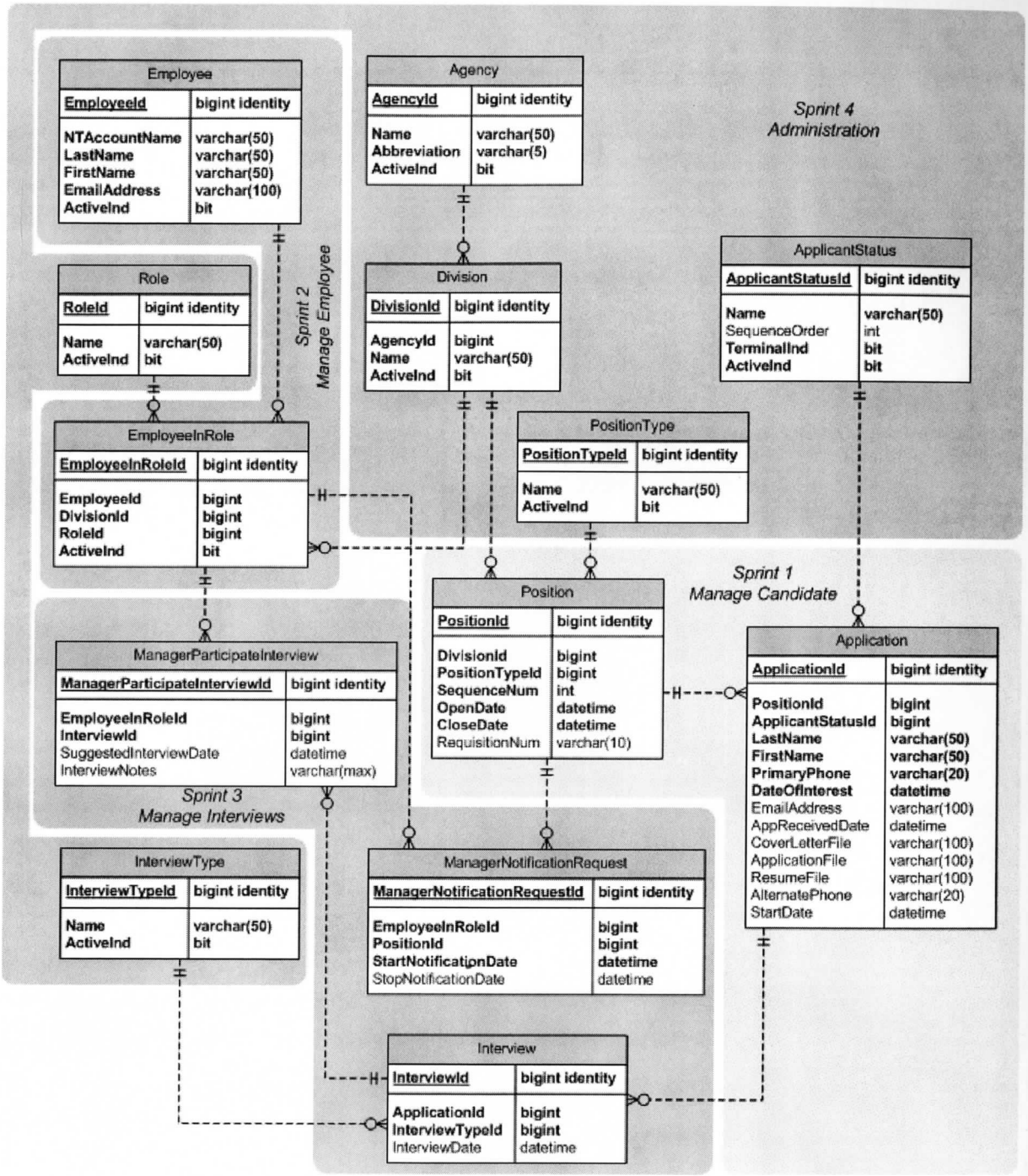
Figure 3 - User Error Example



Database Model

The data model in Figure 3 below will be progressively utilized starting with Sprint 1 through Sprint 4. The entities involved in a Sprint are grouped by colored region. Sprint 4 entities will be utilized in the other Sprints, but will be managed by developers until the administrative interface is developed in Sprint 4.

Figure 4 - Data Model



CHAPTER 3

SPRINT 1 – MANAGE CANDIDATE

Sprint Goals

The goal of the first business sprint is to provide the means to capture candidate information. In order to do this the position information must also be managed. This sprint focuses on opening, listing and closing positions. There are a number of notifications that are also generated in conjunction with position management. The main goal for this sprint is to record candidate information and manage the status from the candidate's initial interest through to either the hiring of the candidate, the candidate declining the offer or management rejecting the candidate.

This sprint only implements rudimentary functionality to allow candidate information to be stored and the status to be manipulated directly without automated notifications. More robust features will be added in a later sprint.

Requirements Information

User Class List

Table 4 – Sprint 1 Expanded User Class List

User Class Name	User Class Description
Recruiting Coordinator	The Recruiting Coordinator is responsible for the overall recruiting process. This person ensures that the candidate data is accurate and facilitates the hiring process within the Division.
BOP Representative	The BOP Representative is one or more people in the Bureau of Personnel that provides information on new candidates and performs the formal communication with the candidates.
Hiring Manager	A manager within the Division that has an opening that they are trying to fill from the pool of prospective candidates.
Division Director	The Division Director periodically reviews candidates, makes determinations regarding marketing efforts based on the number and quality of candidates and adjusts the process as appropriate.

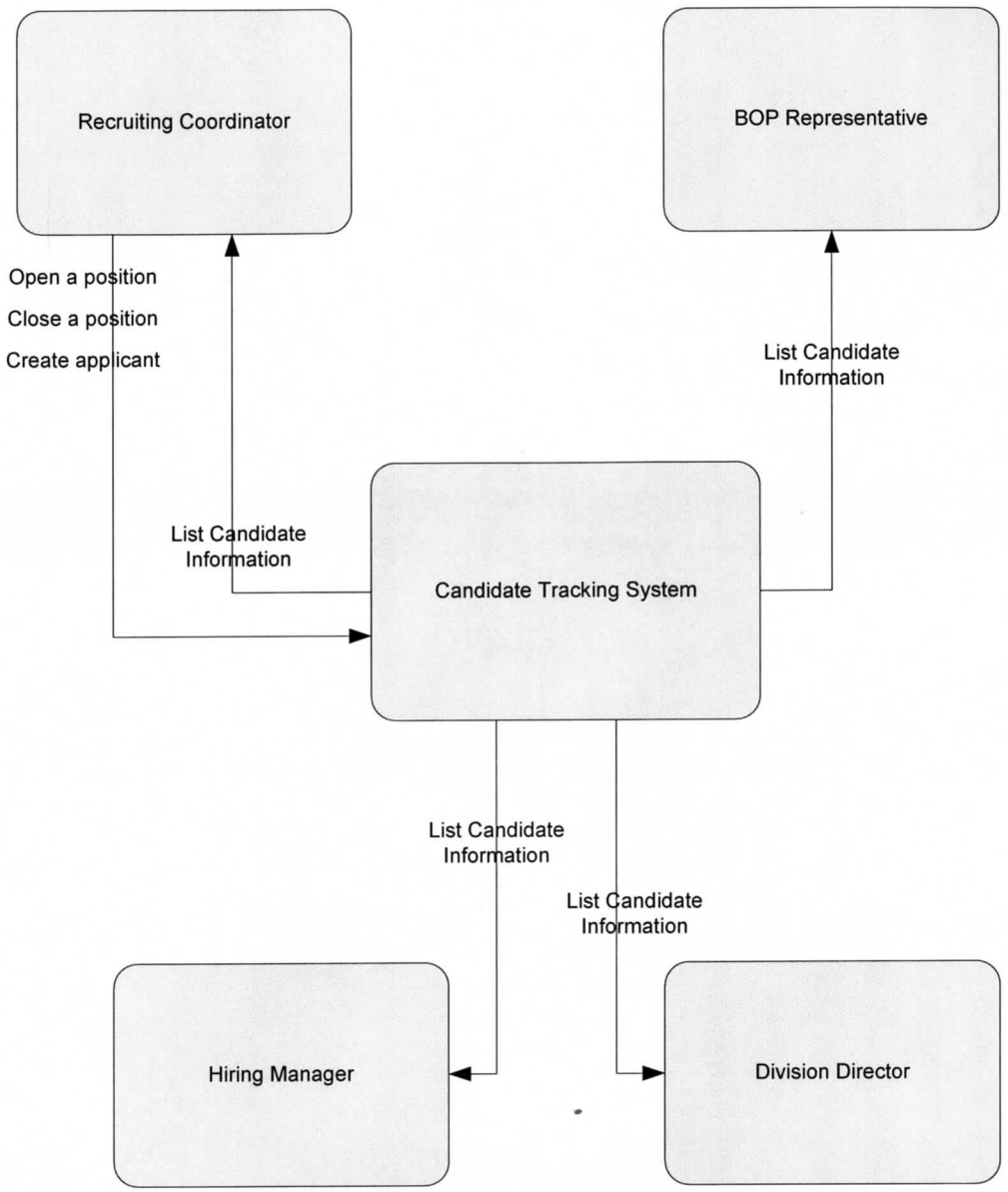
Use Case List with Actors

Table 5 – Sprint 1 Use Case List with Actors

Use Case Id	Actors	Use Case Name
UC-1.1	Recruiting Coordinator	Coordinator opens a position
UC-1.2	Recruiting Coordinator	Coordinator closes a position
UC-1.3	Recruiting Coordinator	Coordinator creates an applicant for a position
UC-1.4	Hiring Manager, Division Director, BOP Representative, Recruiting Coordinator	Everyone lists candidate information
UC-1.5	Hiring Manager, Division Director, BOP Representative, Recruiting Coordinator	Everyone lists position information

Use Case Context Diagram

Figure 5 – Sprint 1 Use Case Context Diagram



Use Case Definitions

Use Case ID:	UC-1.1		
Use Case Name:	Coordinator opens a position		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/11/2009	Date Last Updated:	11/11/2009
Actors:	Recruiting Coordinator		
Description:	The recruiting coordinator opens a position		
Trigger:	The supported division determines the need and resources to open a position.		
Preconditions:	1. The Recruiting Coordinator is authenticated by the system 2. The Agency and Division are available in the system 3. The PositionType is available in the system		
Postconditions:	**Success** 1. All of the position information is saved to the database **Failure** 1. None of the position information is saved to the database		
Normal Flow:	4. The Coordinator enters the position information into the system 5. The Coordinator requests to save the information 6. The system validates the information 7. The system saves the information to the database 8. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 3a. There is invalid data according to BR-1 3a1. The system displays the data that needs to be corrected. [Repeat at 1]		
Exceptions:	1. The desired Position Type is not available. The user will need to contact the system administrator to make the Position Type available until that feature is implemented by the system.		
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:	BR-1: Position Save Rules 1. The open date is a required entry 2. The open date must be equal to or greater than the current date 3. The open date must be less than or equal to the close date 4. The close date must be equal to or greater than the current date		

Use Case ID:	UC-1.2		
Use Case Name:	Coordinator closes a position		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/11/2009	Date Last Updated:	11/11/2009
Actors:	Recruiting Coordinator		
Description:	The recruiting coordinator closes a position		
Trigger:	The supported Division determines an open position should be closed.		
Preconditions:	1. The Recruiting Coordinator is authenticated by the system 2. An open position exists for the Agency and Division		
Postconditions:	**Success** 1. All of the position information is saved to the database 2. The BOP Representative is notified of any active status applicants for the closed position **Failure** 1. No changes are made to the database 2. No notifications are sent		
Normal Flow:	1. The system presents a list of open positions for the Division 2. The Coordinator selects an open position to close 3. The Coordinator indicates to close the selected position 4. The system saves the information to the database 5. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 3a. There are non-terminal status applicants associated with the selected position. 3a1. The system displays a message indicating there are active applicants associated with the selected position 3a2. The system requests confirmation to continue 3a2a: The Coordinator selects to continue 3a2a1. The system notifies the BOP Representative of the active candidates' names and status associated with the position being closed and the position information. 3a2a2. [Continue at 4] 3a2b: The Coordinator selects to discontinue 2a2b1. [Repeat at 1]		
Exceptions			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-1.3		
Use Case Name:	Coordinator creates an applicant		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/11/2009	Date Last Updated:	11/17/2009
Actors:	Recruiting Coordinator		
Description:	The recruiting coordinator creates an applicant and associates the applicant with an open position. The applicant may have application materials and the location in the file system of the application materials is recorded along with the other applicant information. Application materials may be in the form of a PDF, Word or Text file and may contain a resume, application, or cover letter.		
Trigger:	A candidate signifies interest in an open position		
Preconditions:	1. The Recruiting Coordinator is authenticated by the system 2. An open position exists for the Agency and Division for association with the candidate		
Postconditions:	**Success** 1. All of the applicant information is saved to the database 2. New applicants' status is set to "Applied". Existing applicants' status is not updated by this use case. 3. The Hiring Managers are notified of the new applicant with a link to list the applicant information. **Failure** 1. No changes are made to the database 2. No notifications are sent		
Normal Flow:	1. The system presented a list of open positions for the Division 2. The Coordinator selects an open position that the applicant will be associated with 3. The Coordinator enters the applicants information except for status which is set by the system according to BR-2 4. The Coordinator requests to save the applicant information 5. The system validates the information according to BR-3 6. The system saves the information to the database 7. A notification with a link to list the applicant's information is sent to the Hiring Managers 8. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 3a. Update an existing applicant 3a1. The Coordinator requests to update an existing applicant 3a2. The system presents existing applicants for the selected open position. [Continue at 3] 5a. There is invalid data according to BR-2 5a1. The system presents the invalid data [Repeat at 3]		
Exceptions			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:	BR-2: Applicant Status Rules 1. When an applicant is created the status is set to "Applied" 2. If the applicant exists already then the status is not updateable through this use case BR-3: Applicant Save Rules 1. The first name is a required entry 2. The last name is a required entry 3. The date of interest is a required entry 4. An applicant can only apply once to a position (name, primary phone and position constitute a unique combination)		

Use Case ID:	UC-1.4		
Use Case Name:	Everyone lists Candidate Information		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/18/2009	Date Last Updated:	11/18/2009
Actors:	All		
Description:	The user lists applicants for an open position		
Trigger:	A desire to view candidate information		
Preconditions:	1. The user is authenticated by the system 2. The Agency and Division are available in the system 3. One or more open positions exist		
Postconditions:	**Success** 1. The candidate information for the selected position is listed **Failure** 1. No information is listed		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The user selects an open position 3. The system presents a list of applicants who have applied for the selected open position 4. The user selects a candidate for detailed information 5. The system displays the candidate's detailed information		
Alternative Flows:	a. The user quits the operation a1. No information is listed		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-1.5		
Use Case Name:	Everyone lists position information		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	11/18/2009	Date Last Updated:	11/18/2009
Actors:	All		
Description:	The user lists open position information		
Trigger:	A desire to view position information		
Preconditions:	1. The user is authenticated by the system 2. The Agency and Division are available in the system		
Postconditions:	**Success** 1. The open position information is listed **Failure** 1. No information is listed		
Normal Flow:	1. The system presents a list of open positions for the agency and division		
Alternative Flows:	a. The user quits the operation a1. No information is listed		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

CHAPTER 4

SPRINT 2 – MANAGE INTERACTION

Sprint Goals

The goal of the second business sprint is to provide the functionality needed for the interaction between Hiring Managers and Candidates. First is subscription management. This is where a Hiring Manager can subscribe (and unsubscribe) to an open position thus receiving notification when a candidate is created for that position.

When the Recruiting Coordinator creates a candidate in the system those Hiring Managers who are subscribed to the position that the candidate has shown interest in are notified via email that a new candidate is available for review. The Hiring Managers then update the candidate status with either the desire to interview the candidate or a "No thank you" indicating that they are not interested in this candidate. The system tracks interest, or the lack there of, and informs the Recruiting Coordinator via email if there is the need for an interview to be scheduled. If no Hiring Managers are interested in the candidate then the BOP Representative is notified to send a "no thank you" letter to the candidate.

The system tracks the candidate status all the way through start date scheduled sending the appropriate notifications along the way. At any point a candidate's status may change to either "no thank you" or "declined" both of which the system considers a terminal status and sends a notice to the BOP Representative regarding the candidate's status. "Start date scheduled" is also a terminal status and the BOP Representative will be notified.

Requirements Information

User Class List

No changes from previous sprint

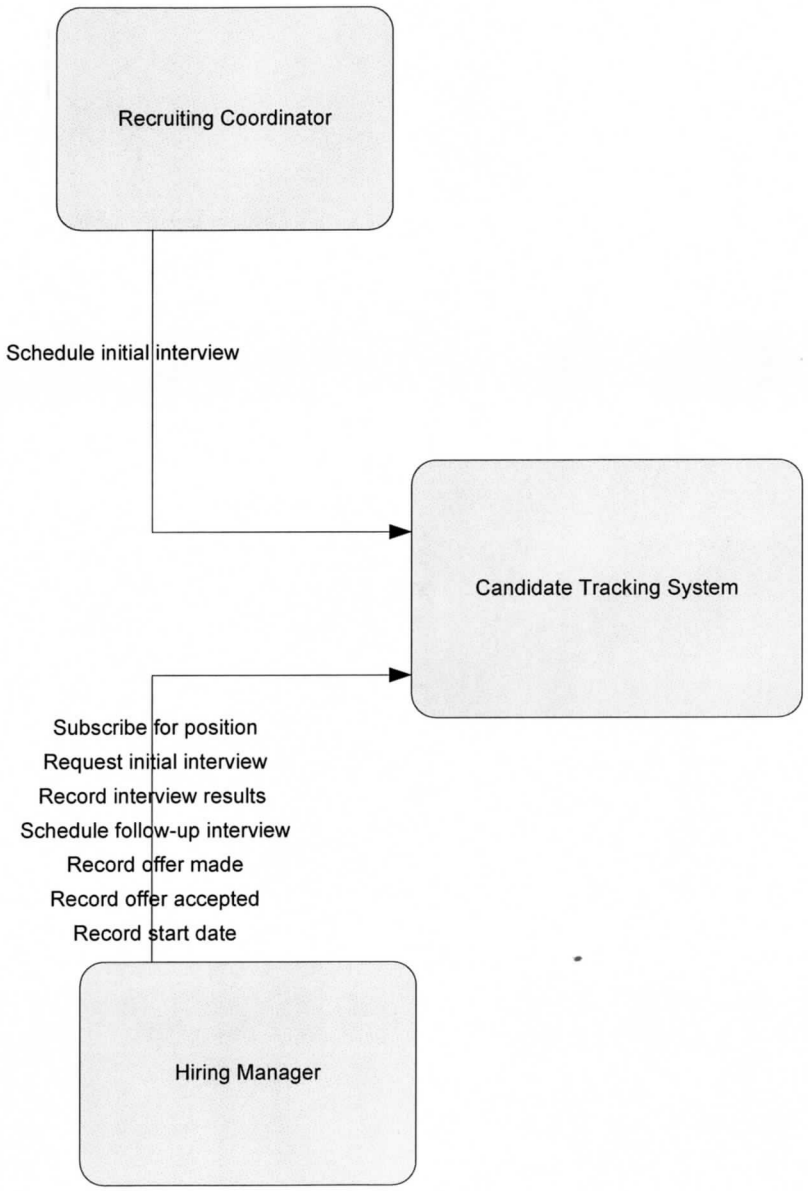
Use Case List with Actors

Table 6 - Sprint 2 Use Case List with Actors

Use Case Id	Actors	Use Case Name
UC-2.1	Hiring Manager	Subscribe for position
UC-2.2	Hiring Manager	Request initial interview
UC-2.3	Recruiting Coordinator	Schedule interview
UC-2.4	Hiring Manager	Record interview results
UC-2.5	Hiring Manager	Schedule follow-up interview
UC-2.6	Hiring Manager	Record offer made
UC-2.7	Hiring Manager	Record offer accepted
UC-2.8	Hiring Manager	Record start date

Use Case Context Diagram

Figure 6 - Sprint 2 Use Case Context Diagram



Use Case Definitions

Use Case ID:	UC-2.1		
Use Case Name:	Hiring Manager subscribe for position		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Hiring Manager		
Description:	The Hiring Manager selects a position to be notified when there is a new applicant.		
Trigger:	A Hiring Manager has an open position and wants to receive notification of new applicants.		
Preconditions:	1. The Hiring Manager is authenticated by the system 2. A position of the type the manager is interested in is open		
Postconditions:	**Success** 1. The manager is configured to receive notifications of applicants for the desired position. **Failure** 1. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Hiring Manager selects an open position for notification 3. The system saves the information to the database 4. The system notifies the Hiring Manager of any applicants currently in the "Applied" status 5. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 2a. The Hiring Manager stops subscribing 2a1. The system presents a list of positions that the Hiring Manager currently has subscriptions to 2a2. The Hiring Manager selects a position to unsubscribe from 2a2. The system discontinues sending notifications for the		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.2		
Use Case Name:	Hiring Manager requests an interview		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	12/30/2009	Date Last Updated:	12/30/2009
Actors:	Hiring Manager		
Description:	The Hiring Manager selects an applicant in the "Applied" status and the system sets the applicant status to "Interview Requested"		
Trigger:	A Hiring Manager has an interest in an applicant in the "Applied" status.		
Preconditions:	1. The Hiring Manager is authenticated by the system 2. One or more applicants are in the "Applied" status		
Postconditions:	**Success** 1. The selected applicant's status is set to "Interview Requested" or it is set to "No thank you" and notifies the BOP Representative **Failure** 1. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Hiring Manager selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Applied" status 4. The Hiring Manager requests to set the status to "Interview Requested" 5. The Hiring Manager requests to save the information 6. The system saves the information to the database 7. The system notifies the Recruiting Coordinator that an interview has been requested 8. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 4a. The Hiring Manager declines an interview 4a0. The Hiring Manager requests to save the information 4a1. The system records the Hiring Manager's decline to interview 4a1a. All subscribed managers have declined to interview 4a1a1. The system notifies the BOP Representative for this agency and division with the applicant's name, position and status		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.3		
Use Case Name:	Recruiting Coordinator schedules interview		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Recruiting Coordinator		
Description:	The Recruiting Coordinator records a suggested interview date between one or more Hiring Managers and a Candidate.		
Trigger:	The Recruiting Coordinator is notified when a Hiring Manager indicates a desire to interview a Candidate.		
Preconditions:	1. The Recruiting Coordinator is authenticated by the system 2. One or more applicants are in the "Interview Requested" status		
Postconditions:	**Success** 1. The Candidate's status is set to "Interview Scheduled" (or "Declined" if the Candidate decides not to pursue the position) 2. The interview date is recorded for the Candidate **Failure** 1. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Coordinator selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Interview Requested" status 4. The Coordinator request to set the status to "Interview Scheduled" 5. The Coordinator requests to save the information 6. The system saves the information to the database 7. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 4a. The Coordinator requests to set the status to "Declined" 4a1. The system sets the status for the applicant to "Declined" 4a2. The system saves the status to the database 4a3. The system notifies the interested Hiring Managers and the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 5]		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.4		
Use Case Name:	Hiring Manager records interview results		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Hiring Manager		
Description:	The Hiring Manager records notes after an interview with a candidate.		
Trigger:	The Hiring Manager completes an interview with a candidate.		
Preconditions:	1. The Hiring manager is authenticated by the system 2. One or more applicants are in the "Interview Scheduled" status		
Postconditions:	**Success** 1. The Hiring Manager's notes are recorded by the system 2. If selected, the candidate's status is set to Declined and the BOP Representative and other Manager's are notified **Failure** 1. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Hiring Manager selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Interview Scheduled" or "Follow-up Interview" status 4. The Hiring Manager records their notes 5. The Hiring Manager requests to schedule a follow-up interview 6. They system sets the candidate status to "Follow-up Interview" 7. The Hiring Manager may enter the follow-up interview date 8. The Hiring Manager requests to save the information 9. The system saves the information to the database 10. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 5a. The Hiring Manager requests to set the status to "Declined" 5a1. The system sets the status for the applicant to "Declined" 5a2. The system saves the status to the database 5a3. The system notifies the interested Hiring Managers and the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 8] 5b. The Manager requests to set the status to "No thank you" 5b1. The system sets the status for the applicant to "No thank you" 5b2. The system saves the status to the database 5b3. The system notifies the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 8]		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.5		
Use Case Name:	Hiring Manager schedules a follow-up interview		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Hiring Manager		
Description:	The Hiring Manager records an interview date between one or more Hiring Managers and a Candidate.		
Trigger:	The Hiring Manager has a desire to do another interview with a Candidate.		
Preconditions:	1. The Hiring Manager is authenticated by the system		
Postconditions:	**Success** 1. The Candidate's status is set to "Follow-up Interview" (or "Declined" if the Candidate decides not to pursue the position or "No Thank You" if the Hiring Manager is no longer interested) 2. The interview date is recorded for the Candidate **Failure** 2. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Coordinator selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Interview Scheduled" or "Follow-up Interview" status 4. The Hiring Manager request to set the status to "Follow-up Interview" 5. The Hiring Manager enters the Interview Date 6. The Hiring Manager requests to save the information 7. The system saves the information to the database 8. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 4a. The Manager requests to set the status to "No thank you" 4a1. The system sets the status for the applicant to "No thank you" 4a2. The system saves the status to the database 4a3. The system notifies the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 6] 4b. The Manager requests to set the status to "Declined" 4b1. The system sets the status for the applicant to "Declined" 4b2. The system saves the status to the database 4b3. The system notifies the interested Hiring Managers and the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 6]		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.6		
Use Case Name:	Hiring Manager records offer made		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Hiring Manager		
Description:	The Hiring Manager records that an offer has been made to a Candidate.		
Trigger:	The Hiring Manager desires to make an offer to a Candidate.		
Preconditions:	1. The Hiring Manager is authenticated by the system		
Postconditions:	**Success** 1. The Candidate's status is set to "Make Offer" (or "Declined" if the Candidate decides not to pursue the position or "No Thank You" if the Hiring Manager is no longer interested) **Failure** 1. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Coordinator selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Interview Scheduled" or "Follow-up Interview" status 4. The Hiring Manager request to set the status to "Make Offer" 5. The Hiring Manager requests to save the information 6. The system saves the information to the database 7. The system displays a confirmation of the changes		
Alternative Flows:	b. The user quits the operation a1. No changes are saved to the database 4a. The Manager requests to set the status to "No thank you" 4a1. The system sets the status for the applicant to "No thank you" 4a2. The system saves the status to the database 4a3. The system notifies the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 5] 4b. The Manager requests to set the status to "Declined" 4b1. The system sets the status for the applicant to "Declined" 4b2. The system saves the status to the database 4b3. The system notifies the interested Hiring Managers and the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 5]		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.7		
Use Case Name:	Hiring Manager records offer accepted		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Hiring Manager		
Description:	The Hiring Manager records that an offer was accepted by a Candidate.		
Trigger:	The Hiring Manager hears from a candidate that they have accepted the offer.		
Preconditions:	1. The Hiring Manager is authenticated by the system		
Postconditions:	**Success** 1. The Candidate's status is set to "Offer Accepted" (or "Declined" if the Candidate decides not to pursue the position or "No Thank You" if the Hiring Manager is no longer interested) **Failure** 2. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Coordinator selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Make Offer" status 4. The Hiring Manager request to set the status to "Offer Accepted" 5. The Hiring Manager requests to save the information 6. The system saves the information to the database 7. The system displays a confirmation of the changes		
Alternative Flows:	a. The user quits the operation a1. No changes are saved to the database 4a. The Manager requests to set the status to "No thank you" 4a1. The system sets the status for the applicant to "No thank you" 4a2. The system saves the status to the database 4a3. The system notifies the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 5] 4b. The Manager requests to set the status to "Declined" 4b1. The system sets the status for the applicant to "Declined" 4b2. The system saves the status to the database 4b3. The system notifies the interested Hiring Managers and the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 5]		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

Use Case ID:	UC-2.8		
Use Case Name:	Hiring Manager records start date		
Created By:	Dave Bishop	Last Updated By:	Dave Bishop
Date Created:	1/1/2010	Date Last Updated:	1/1/2010
Actors:	Hiring Manager		
Description:	The Hiring Manager records a start date for a candidate.		
Trigger:	The Hiring Manager and candidate have determined a start date.		
Preconditions:	1. The Hiring Manager is authenticated by the system		
Postconditions:	**Success** 1. The Candidate's status is set to "Start date scheduled" (or "Declined" if the Candidate decides not to pursue the position or "No Thank You" if the Hiring Manager is no longer interested) 2. The BOP Representative and Division Director are notified of the candidate's status and start date **Failure** 1. No changes are made to the database		
Normal Flow:	1. The system presents a list of open positions for the agency and division 2. The Coordinator selects an open position 3. The system presents a list of applicants who have applied for the selected open position and are in the "Offer Accepted" status 4. The Hiring Manager request to set the status to "Start date scheduled" 5. The Hiring Manager enters the scheduled start date 6. The Hiring Manager requests to save the information 7. The BOP Representative and Division Director are notified of the candidate's status and start date 8. The system saves the information to the database 9. The system displays a confirmation of the changes		
Alternative Flows:	b. The user quits the operation a1. No changes are saved to the database 4a. The Manager requests to set the status to "No thank you" 4a1. The system sets the status for the applicant to "No thank you" 4a2. The system saves the status to the database 4a3. The system notifies the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 8] 4b. The Manager requests to set the status to "Declined" 4b1. The system sets the status for the applicant to "Declined" 4b2. The system saves the status to the database 4b3. The system notifies the interested Hiring Managers and the BOP Representative for this agency and division with the applicant's name, position and status [Continue at 8]		
Exceptions:			
Includes:			
Priority:	High		
Frequency of Use:	High		
Business Rules:			

CHAPTER 5

LESSONS LEARNED

Project Management with Scrum

Delivery order maximizes value

Reflecting on this effort compared to another recent project revealed an import lesson on delivering value to the sponsor. In the other project we chose to implement the administrative features first. By administrative features we mean the functionality to maintain values for various system tables that populate dropdown lists, adding users to the system and maintaining roles for the users. In the Candidate Tracking System the decision was made to implement functionality that benefited the sponsor's goals first. The last sprint will handle the administrative needs. In the meantime the necessary values are populated with SQL seed data scripts and the few changes required will be handled by the programmer interactively.

This allows the project to deliver value to the sponsor immediately rather than delay while the administrative features are defined and implemented. Although the first sprint had minimal features, the few features supported useful business processes. Each subsequent sprint builds on the business features. Only after the priority business features have been implemented will the administrative sprint be accomplished to include the features necessary to maintain system data and user access.

One of the reasons this approach worked on the Candidate Tracking System is that it is initially intended for the BIT Development Division which has seven Managers, one Director and one BOP Representative. This made the user access management reasonable to accomplish using seed data and manual SQL. If the audience was significantly larger the user management features would need to be built sooner. The sponsor was clear that she wanted the features to work in the BIT Division first, and later, be able to extend the audience to other divisions and agencies. This fit well with the value first proposition of Scrum.

Trade-offs in a time-boxed environment

Scrum's emphasis on delivering value on a time-boxed schedule helped drive the decisions throughout the project. There were many points along the way where a more interesting user interface was possible, but since there was such a short timeframe to deliver the functionality the decision was made to go with the simplest approach that would deliver the desired functionality.

In other projects with less of a sense of urgency we often experiment with various options when any one of them would have delivered the feature. The short duration of a sprint kept the goal in focus and helped cut through the fog of options and potential implementations. It produced a laser focus on getting the feature implemented because the sprint deadline was looming. The selection of a reasonable sized set of features for the sprint and the specific timelines were exceedingly helpful in getting this project completed on time.

One example is the date picking user interface. The Visual Studio 2008 development environment comes with a build in Calendar user interface element. It displays as a full calendar. It is entirely possible to use an image of a calendar and when the image is clicked to display the calendar, allow the user to pick the date from the calendar and copy the date to a text field then hide the calendar. This would have been a nice enhancement to the user interface, but was it necessary? No. The decision was to forge ahead and get the features implemented without the "fancy" interface element.

Later, if the sponsor requests it, the calendar can be enhanced, but the current implementation meets the requirement of choosing a date graphically, minimizing the potential of incorrect date format issues.

The value of a data access code generator

Another great discover on this project was Scott Darnall's code generator. Scott developed an application that uses the SQL Server database as the input to generate a set of coordinated stored procedures, VB.Net data classes, VB.Net collection classes and VB.Net business classes. There are a few simple naming conventions and index definitions to maximize the value of the code generator. Once these are in place just press a button and the generator creates code to incorporate into your project to simplify typical create, read, update and delete (CRUD) scenarios.

The code generator crated over 2,000 lines of VB.Net class code to manage data access for the Candidate Tracking System. It also generated thousands of lines of stored procedure code. This enabled the developer to write a statement like Candidate.Save() and the generated Candidate business object would handle all of the database interaction and include some simple integrity checks for data type and optionality.

This was a huge time saver for the project, many thanks to Scott Darnall for this valuable tool!

The difficulty of artifact maintenance

One final lesson learned, or at least reinforced, on this project: The difficulty of creating and maintaining requirements, design and test artifacts. Executing the project in smaller sprints helped somewhat to minimize the amount of artifacts necessary to be generated and updated throughout the sprint. But even a small amount of artifacts becomes annoying, especially when the project involves a single human resource.

My experience was that the requirements and design documents were valuable at the start of the sprint. They helped organize and focus attention and ensure a well thought out set of features. As the sprint unfolded and the code started to come together decisions were made that modified what was written in some of the use cases or in the user interface designs. I found myself writing the code and not seeing the value of going back and updating the use cases or user interface design documents to match what was built.

As a Development Manager and Project Manager I have often been on the side that is requesting a team to keep all the artifacts up-to-date and now I can at least better sympathize with the Developer who questions the value of doing so. I recognize that the artifacts are valuable when someone else needs visibility into the project, to get an overview of the application, to create test cases or to create user guides. These are just difficult to maintain from a sole developer perspective in the heat of the battle that is application development.

APPENDICES

APPENDIX A: DATABASE CODE EXAMPLES

A generated stored procedure - spsApplication

```
ALTER PROCEDURE [dbo].[spsApplication] (@pbiApplicationId bigint,
@pbiPositionId bigint, @psLastName varchar(50), @psFirstName varchar(50),
@psPrimaryPhone varchar(20), @psAlternatePhone varchar(20), @psEmailAddress
varchar(100), @pbiApplicantStatusId bigint, @pdDateOfInterest datetime,
@pdAppReceivedDate datetime, @psCoverLetterFile varchar(300),
@psApplicationFile varchar(300), @psResumeFile varchar(300), @pdStartDate
datetime, @piID int output) AS
IF @pbiApplicationId > 0
      BEGIN
            UPDATE Application SET
            PositionId = @pbiPositionId,
            LastName = @psLastName,
            FirstName = @psFirstName,
            PrimaryPhone = @psPrimaryPhone,
            AlternatePhone = @psAlternatePhone,
            EmailAddress = @psEmailAddress,
            ApplicantStatusId = @pbiApplicantStatusId,
            DateOfInterest = @pdDateOfInterest,
            AppReceivedDate = @pdAppReceivedDate,
            CoverLetterFile = @psCoverLetterFile,
            ApplicationFile = @psApplicationFile,
            ResumeFile = @psResumeFile,
            StartDate = @pdStartDate
            WHERE ApplicationId = @pbiApplicationId
            SELECT @piID = @pbiApplicationId
      END
ELSE
      BEGIN
            INSERT INTO Application(PositionId, LastName, FirstName,
PrimaryPhone, AlternatePhone, EmailAddress, ApplicantStatusId,
DateOfInterest, AppReceivedDate, CoverLetterFile, ApplicationFile,
ResumeFile, StartDate)
            VALUES (@pbiPositionId, @psLastName, @psFirstName,
@psPrimaryPhone, @psAlternatePhone, @psEmailAddress, @pbiApplicantStatusId,
@pdDateOfInterest, @pdAppReceivedDate, @psCoverLetterFile,
@psApplicationFile, @psResumeFile, @pdStartDate)
            SELECT @piID = @@Identity
      END
```

A custom stored procedure -

custGetInterestedManagerCountByApplicationIdPositionId

```
-- ============================================
-- Author:        Dave Bishop
-- Create date: 1/12/2010
-- ============================================
ALTER PROCEDURE
[dbo].[custGetInterestedManagerCountByApplicationIdPositionId]
      -- Add the parameters for the stored procedure here
      @pbiApplicationId BIGINT,
      @pbiPositionId BIGINT
AS
DECLARE
@Now DATETIME
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON
    SELECT @Now = GetDate()

      SELECT COUNT(ManagerNotificationRequestId)ManagerCount
      FROM ManagerNotificationRequest mnr
      WHERE PositionId = @pbiPositionId
      AND StartNotificationDate <= @Now
      AND ISNULL(StopNotificationDate,@Now) >= @Now
      AND NOT EXISTS (SELECT ManagerInterestInterviewId
                        FROM ManagerInterestInterview mii
                        WHERE mii.EmployeeInRoleId =
mnr.EmployeeInRoleId
                            AND mii.ApplicationId = @pbiApplicationId
                            AND mii.DeclineDate IS NOT NULL)
END
```

APPENDIX B: MIDDLE TIER CODE EXAMPLES

A generated data class - CDApplication

```
Option Explicit On
Option Strict On

Public MustInherit Class CDApplication
      Inherits CAdmin

      ' Data Table Variables
      Private cbiApplicationId As Int64
      Private cbiPositionId As Int64
      Private csLastName As String
      Private csFirstName As String
      Private csPrimaryPhone As String
      Private csAlternatePhone As String
      Private csEmailAddress As String
      Private cbiApplicantStatusId As Int64
      Private cdDateOfInterest As Date
      Private cdAppReceivedDate As Date
      Private csCoverLetterFile As String
      Private csApplicationFile As String
      Private csResumeFile As String
      Private cdStartDate As Date

      Private cbModified As Boolean

      Public Sub New(ByVal psUser As String)
            ' Used for new records

            Try

                  cbiApplicationId = 0
                  cbModified = False

            Catch appException As ApplicationException
                  Throw

            Catch uhException As Exception
                  Throw

            End Try

      End Sub

      Public Sub New(ByVal psUser As String, ByVal pbiApplicationId AS
Int64)

            Dim lcmdApplication As SqlClient.SqlCommand
            Dim lparApplicationId As SqlClient.SqlParameter
```

```
            Try
                  lcmdApplication = New SqlClient.SqlCommand()
                  lcmdApplication.CommandType = CommandType.StoredProcedure
                  lcmdApplication.CommandText =
"spqApplication_ApplicationId"

                  lparApplicationId =
lcmdApplication.Parameters.Add("@pbiApplicationId", SQLDBType.bigint)
                  lparApplicationId.Value = pbiApplicationId

                  LoadValues(lcmdApplication)

                  cbModified = False

            Catch sqlException As SQLClient.SQLException
                  Throw

            Catch appException As ApplicationException
                  Throw

            Catch uhException As Exception
                  Throw

            Finally

                  lcmdApplication.Dispose()
                  lcmdApplication = Nothing

            End Try
      End Sub

      Public Sub New(ByVal psUser As String, ByVal pbiPositionId AS Int64,
ByVal psLastName AS String, ByVal psFirstName AS String, ByVal
psPrimaryPhone AS String)

            Dim lcmdApplication As SqlClient.SqlCommand
            Dim lparPositionId As SqlClient.SqlParameter
            Dim lparLastName As SqlClient.SqlParameter
            Dim lparFirstName As SqlClient.SqlParameter
            Dim lparPrimaryPhone As SqlClient.SqlParameter

            Try
                  lcmdApplication = New SqlClient.SqlCommand()
                  lcmdApplication.CommandType = CommandType.StoredProcedure
                  lcmdApplication.CommandText =
"spqApplication_PositionIdLastNameFirstNamePrimaryPhone"

                  lparPositionId =
lcmdApplication.Parameters.Add("@pbiPositionId", SQLDBType.bigint)
                  lparPositionId.Value = pbiPositionId
                  lparLastName =
lcmdApplication.Parameters.Add("@psLastName", SQLDBType.varchar)
                  lparLastName.Value = psLastName
                  lparFirstName =
lcmdApplication.Parameters.Add("@psFirstName", SQLDBType.varchar)
                  lparFirstName.Value = psFirstName
```

```
                lparPrimaryPhone =
lcmdApplication.Parameters.Add("@psPrimaryPhone", SQLDBType.varchar)
                lparPrimaryPhone.Value = psPrimaryPhone

                LoadValues(lcmdApplication)

                cbModified = False

            Catch sqlException As SQLClient.SQLException
                Throw

            Catch appException As ApplicationException
                Throw

            Catch uhException As Exception
                Throw

            Finally

                lcmdApplication.Dispose()
                lcmdApplication = Nothing

            End Try
        End Sub

        Private Sub LoadValues(ByVal pcmdApplication As SqlClient.SqlCommand)
            ' Loads record values into class

            Dim lsdrApplication As SqlClient.SqlDataReader
            Dim lscnApplication As SqlClient.SqlConnection

            Try
                lscnApplication = New SqlClient.SqlConnection()
                lscnApplication.ConnectionString = Me.ConnectionString
                lscnApplication.Open()
                pcmdApplication.Connection = lscnApplication

                lsdrApplication = pcmdApplication.ExecuteReader()
                If lsdrApplication.Read Then
                    Try
                        cbiApplicationId =
lsdrApplication.GetInt64(0)
                    Catch
                        cbiApplicationId = Nothing
                    End Try
                    Try
                        cbiPositionId = lsdrApplication.GetInt64(1)
                    Catch
                        cbiPositionId = Nothing
                    End Try
                    Try
                        csLastName = lsdrApplication.GetString(2)
                    Catch
                        csLastName = Nothing
                    End Try
                    Try
                        csFirstName = lsdrApplication.GetString(3)
```

```
                        Catch
                              csFirstName = Nothing
                        End Try
                        Try
                              csPrimaryPhone = lsdrApplication.GetString(4)
                        Catch
                              csPrimaryPhone = Nothing
                        End Try
                        Try
                              csAlternatePhone =
lsdrApplication.GetString(5)
                        Catch
                              csAlternatePhone = Nothing
                        End Try
                        Try
                              csEmailAddress = lsdrApplication.GetString(6)
                        Catch
                              csEmailAddress = Nothing
                        End Try
                        Try
                              cbiApplicantStatusId =
lsdrApplication.GetInt64(7)
                        Catch
                              cbiApplicantStatusId = Nothing
                        End Try
                        Try
                              cdDateOfInterest =
lsdrApplication.GetDateTime(8)
                        Catch
                              cdDateOfInterest = Nothing
                        End Try
                        Try
                              cdAppReceivedDate =
lsdrApplication.GetDateTime(9)
                        Catch
                              cdAppReceivedDate = Nothing
                        End Try
                        Try
                              csCoverLetterFile =
lsdrApplication.GetString(10)
                        Catch
                              csCoverLetterFile = Nothing
                        End Try
                        Try
                              csApplicationFile =
lsdrApplication.GetString(11)
                        Catch
                              csApplicationFile = Nothing
                        End Try
                        Try
                              csResumeFile = lsdrApplication.GetString(12)
                        Catch
                              csResumeFile = Nothing
                        End Try
                        Try
                              cdStartDate = lsdrApplication.GetDateTime(13)
                        Catch
```

```
                        cdStartDate = Nothing
                    End Try
                Else
                    Throw New System.ApplicationException("No record
found.")
                End If

            Catch appException As ApplicationException
                Throw

            Catch sqlException As SQLClient.SQLException
                Throw

            Catch uhException As Exception
                Throw

            Finally
                lscnApplication.Close()
                lscnApplication.Dispose
                lsdrApplication.Close()
                lsdrApplication = Nothing
                pcmdApplication = Nothing
            End Try

        End Sub

        Public Sub Save()

            Dim lscnApplication As SqlClient.SqlConnection
            Dim lcmdApplication As SqlClient.SqlCommand
            Dim lparApplicationId As SqlClient.SqlParameter
            Dim lparPositionId As SqlClient.SqlParameter
            Dim lparLastName As SqlClient.SqlParameter
            Dim lparFirstName As SqlClient.SqlParameter
            Dim lparPrimaryPhone As SqlClient.SqlParameter
            Dim lparAlternatePhone As SqlClient.SqlParameter
            Dim lparEmailAddress As SqlClient.SqlParameter
            Dim lparApplicantStatusId As SqlClient.SqlParameter
            Dim lparDateOfInterest As SqlClient.SqlParameter
            Dim lparAppReceivedDate As SqlClient.SqlParameter
            Dim lparCoverLetterFile As SqlClient.SqlParameter
            Dim lparApplicationFile As SqlClient.SqlParameter
            Dim lparResumeFile As SqlClient.SqlParameter
            Dim lparStartDate As SqlClient.SqlParameter

            Dim lparNewID As SqlClient.SqlParameter
            Dim lsdrApplication As SqlClient.SqlDataReader

            Try
                lscnApplication = New SqlClient.SqlConnection()
                lscnApplication.ConnectionString = Me.ConnectionString
                lscnApplication.Open()
                lcmdApplication = New SqlClient.SqlCommand()
                lcmdApplication.CommandType = CommandType.StoredProcedure
                lcmdApplication.CommandText = "spsApplication"
                lcmdApplication.Connection = lscnApplication
```

```
                lparApplicationId =
lcmdApplication.Parameters.Add("@pbiApplicationId", SQLDBType.bigint)
                lparApplicationId.Value = Me.ApplicationId
                lparPositionId =
lcmdApplication.Parameters.Add("@pbiPositionId", SQLDBType.bigint)
                If cbiPositionId = Nothing Then
                      Throw New System.ApplicationException("PositionId
is required.")
                Else
                      lparPositionId.Value = Me.PositionId
                End If
                lparLastName =
lcmdApplication.Parameters.Add("@psLastName", SQLDBType.varchar)
                If csLastName = Nothing Then
                      Throw New System.ApplicationException("LastName is
required.")
                Else
                      lparLastName.Value = Me.LastName
                End If
                lparFirstName =
lcmdApplication.Parameters.Add("@psFirstName", SQLDBType.varchar)
                If csFirstName = Nothing Then
                      Throw New System.ApplicationException("FirstName is
required.")
                Else
                      lparFirstName.Value = Me.FirstName
                End If
                lparPrimaryPhone =
lcmdApplication.Parameters.Add("@psPrimaryPhone", SQLDBType.varchar)
                If csPrimaryPhone = Nothing Then
                      Throw New System.ApplicationException("PrimaryPhone
is required.")
                Else
                      lparPrimaryPhone.Value = Me.PrimaryPhone
                End If
                lparAlternatePhone =
lcmdApplication.Parameters.Add("@psAlternatePhone", SQLDBType.varchar)
                If csAlternatePhone = Nothing Then
                      lparAlternatePhone.Value = DBNull.Value
                Else
                      lparAlternatePhone.Value = Me.AlternatePhone
                End If
                lparEmailAddress =
lcmdApplication.Parameters.Add("@psEmailAddress", SQLDBType.varchar)
                If csEmailAddress = Nothing Then
                      lparEmailAddress.Value = DBNull.Value
                Else
                      lparEmailAddress.Value = Me.EmailAddress
                End If
                lparApplicantStatusId =
lcmdApplication.Parameters.Add("@pbiApplicantStatusId", SQLDBType.bigint)
                If cbiApplicantStatusId = Nothing Then
                      Throw New
System.ApplicationException("ApplicantStatusId is required.")
                Else
                      lparApplicantStatusId.Value = Me.ApplicantStatusId
                End If
```

```
                lparDateOfInterest =
lcmdApplication.Parameters.Add("@pdDateOfInterest", SQLDBType.datetime)
                If cdDateOfInterest = Nothing Then
                      Throw New
System.ApplicationException("DateOfInterest is required.")
                Else
                      lparDateOfInterest.Value = Me.DateOfInterest
                End If
                lparAppReceivedDate =
lcmdApplication.Parameters.Add("@pdAppReceivedDate", SQLDBType.datetime)
                If cdAppReceivedDate = Nothing Then
                      lparAppReceivedDate.Value = DBNull.Value
                Else
                      lparAppReceivedDate.Value = Me.AppReceivedDate
                End If
                lparCoverLetterFile =
lcmdApplication.Parameters.Add("@psCoverLetterFile", SQLDBType.varchar)
                If csCoverLetterFile = Nothing Then
                      lparCoverLetterFile.Value = DBNull.Value
                Else
                      lparCoverLetterFile.Value = Me.CoverLetterFile
                End If
                lparApplicationFile =
lcmdApplication.Parameters.Add("@psApplicationFile", SQLDBType.varchar)
                If csApplicationFile = Nothing Then
                      lparApplicationFile.Value = DBNull.Value
                Else
                      lparApplicationFile.Value = Me.ApplicationFile
                End If
                lparResumeFile =
lcmdApplication.Parameters.Add("@psResumeFile", SQLDBType.varchar)
                If csResumeFile = Nothing Then
                      lparResumeFile.Value = DBNull.Value
                Else
                      lparResumeFile.Value = Me.ResumeFile
                End If
                lparStartDate =
lcmdApplication.Parameters.Add("@pdStartDate", SQLDBType.datetime)
                If cdStartDate = Nothing Then
                      lparStartDate.Value = DBNull.Value
                Else
                      lparStartDate.Value = Me.StartDate
                End If

                lparNewID = lcmdApplication.Parameters.Add("@piID",
SqlDbType.Int)
                lparNewID.Direction = ParameterDirection.Output

                lsdrApplication = lcmdApplication.ExecuteReader()
                cbiApplicationId =
CType(lcmdApplication.Parameters("@piID").Value(),Int64)
                lsdrApplication.Close()
                cbModified = False

          Catch sqlException As SQLClient.SQLException
                Throw
```

```
            Catch appException As ApplicationException
                  Throw

            Catch uhException As Exception
                  Throw

            Finally

                  lsdrApplication = Nothing
                  lcmdApplication.Dispose()
                  lcmdApplication = Nothing
                  lscnApplication.Close()
                  lscnApplication = Nothing

            End Try

      End Sub

      Public Sub Delete()

            Dim lscnApplication As SqlClient.SqlConnection
            Dim lcmdApplication As SqlClient.SqlCommand
            Dim lparApplicationId As SqlClient.SqlParameter
            Dim lsdrApplication As SqlClient.SqlDataReader

            Try
                        lscnApplication = New SqlClient.SqlConnection()
                        lscnApplication.ConnectionString =
Me.ConnectionString
                        lscnApplication.Open()
                        lcmdApplication = New SqlClient.SqlCommand()
                        lcmdApplication.CommandType =
CommandType.StoredProcedure
                        lcmdApplication.CommandText = "spdApplication"
                        lcmdApplication.Connection = lscnApplication

                        lparApplicationId =
lcmdApplication.Parameters.Add("@pbiApplicationId", SqlDbType.bigint)
                        lparApplicationId.Value = Me.ApplicationId

                        lsdrApplication = lcmdApplication.ExecuteReader()

            Catch appException As ApplicationException
                  Throw

            Catch sqlException As SqlClient.SqlException
                  Throw

            Catch uhException As Exception
                  Throw

            Finally

                  lsdrApplication.Close()
                  lsdrApplication = Nothing
                  lcmdApplication.Dispose()
                  lcmdApplication = Nothing
```

```
                lscnApplication.Close()
                lscnApplication = Nothing

            End Try
        End Sub

        Public ReadOnly Property ApplicationId As Int64
            Get
                Try
                    ApplicationId = cbiApplicationId
                Catch uhException As Exception
                    Throw
                End Try
            End Get
        End Property

        Public Property PositionId As Int64
            Get
                Try
                    PositionId = cbiPositionId
                Catch uhException As Exception
                    Throw
                End Try
            End Get
            Set(ByVal Value As Int64)
                Try
                    If Value.Equals(Nothing) Then
                        Throw New
System.ApplicationException("PositionId is required.")
                    End If
                    cbiPositionId = Value
                    cbModified = True
                Catch appException As ApplicationException
                    Throw
                Catch uhException As Exception
                    Throw
                End Try
            End Set
        End Property

        Public Property LastName As String
            Get
                Try
                    LastName = csLastName
                Catch uhException As Exception
                    Throw
                End Try
            End Get
            Set(ByVal Value As String)
                Try
                    If Value.Equals(Nothing) Then
                        Throw New
System.ApplicationException("LastName is required.")
                    End If
                    If Value.Length > 50 Then
                        Throw New
System.ApplicationException("LastName must be 50 Characters or less.")
```

```
                        End If
                        csLastName = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
      End Property

      Public Property FirstName As String
            Get
                  Try
                        FirstName = csFirstName
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As String)
                  Try
                        If Value.Equals(Nothing) Then
                              Throw New
System.ApplicationException("FirstName is required.")
                        End If
                        If Value.Length > 50 Then
                              Throw New
System.ApplicationException("FirstName must be 50 Characters or less.")
                        End If
                        csFirstName = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
      End Property

      Public Property PrimaryPhone As String
            Get
                  Try
                        PrimaryPhone = csPrimaryPhone
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As String)
                  Try
                        If Value.Equals(Nothing) Then
                              Throw New
System.ApplicationException("PrimaryPhone is required.")
                        End If
                        If Value.Length > 20 Then
                              Throw New
System.ApplicationException("PrimaryPhone must be 20 Characters or less.")
                        End If
```

```
                        csPrimaryPhone = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
      End Property

      Public Property AlternatePhone As String
            Get
                  Try
                        AlternatePhone = csAlternatePhone
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As String)
                  Try
                        If Value.Length > 20 Then
                              Throw New
System.ApplicationException("AlternatePhone must be 20 Characters or
less.")
                        End If
                        csAlternatePhone = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
      End Property

      Public Property EmailAddress As String
            Get
                  Try
                        EmailAddress = csEmailAddress
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As String)
                  Try
                        If Value.Length > 100 Then
                              Throw New
System.ApplicationException("EmailAddress must be 100 Characters or less.")
                        End If
                        csEmailAddress = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
```

```
        End Property

        Public Property ApplicantStatusId As Int64
            Get
                Try
                    ApplicantStatusId = cbiApplicantStatusId
                Catch uhException As Exception
                    Throw
                End Try
            End Get
            Set(ByVal Value As Int64)
                Try
                    If Value.Equals(Nothing) Then
                        Throw New
System.ApplicationException("ApplicantStatusId is required.")
                    End If
                    cbiApplicantStatusId = Value
                    cbModified = True
                Catch appException As ApplicationException
                    Throw
                Catch uhException As Exception
                    Throw
                End Try
            End Set
        End Property

        Public Property DateOfInterest As Date
            Get
                Try
                    DateOfInterest = cdDateOfInterest
                Catch uhException As Exception
                    Throw
                End Try
            End Get
            Set(ByVal Value As Date)
                Try
                    If Value.Equals(Nothing) Then
                        Throw New
System.ApplicationException("DateOfInterest is required.")
                    End If
                    cdDateOfInterest = Value
                    cbModified = True
                Catch appException As ApplicationException
                    Throw
                Catch uhException As Exception
                    Throw
                End Try
            End Set
        End Property

        Public Property AppReceivedDate As Date
            Get
                Try
                    AppReceivedDate = cdAppReceivedDate
                Catch uhException As Exception
                    Throw
                End Try
```

```
            End Get
            Set(ByVal Value As Date)
                  Try
                        cdAppReceivedDate = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
      End Property

      Public Property CoverLetterFile As String
            Get
                  Try
                        CoverLetterFile = csCoverLetterFile
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As String)
                  Try
                        If Value.Length > 300 Then
                              Throw New
System.ApplicationException("CoverLetterFile must be 300 Characters or
less.")
                        End If
                        csCoverLetterFile = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
                  Catch uhException As Exception
                        Throw
                  End Try
            End Set
      End Property

      Public Property ApplicationFile As String
            Get
                  Try
                        ApplicationFile = csApplicationFile
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As String)
                  Try
                        If Value.Length > 300 Then
                              Throw New
System.ApplicationException("ApplicationFile must be 300 Characters or
less.")
                        End If
                        csApplicationFile = Value
                        cbModified = True
                  Catch appException As ApplicationException
                        Throw
```

```
                Catch uhException As Exception
                      Throw
                End Try
          End Set
    End Property

    Public Property ResumeFile As String
          Get
                Try
                      ResumeFile = csResumeFile
                Catch uhException As Exception
                      Throw
                End Try
          End Get
          Set(ByVal Value As String)
                Try
                      If Value.Length > 300 Then
                            Throw New
System.ApplicationException("ResumeFile must be 300 Characters or less.")
                      End If
                      csResumeFile = Value
                      cbModified = True
                Catch appException As ApplicationException
                      Throw
                Catch uhException As Exception
                      Throw
                End Try
          End Set
    End Property

    Public Property StartDate As Date
          Get
                Try
                      StartDate = cdStartDate
                Catch uhException As Exception
                      Throw
                End Try
          End Get
          Set(ByVal Value As Date)
                Try
                      cdStartDate = Value
                      cbModified = True
                Catch appException As ApplicationException
                      Throw
                Catch uhException As Exception
                      Throw
                End Try
          End Set
    End Property

    Public ReadOnly Property Modified As Boolean
          Get
                Try
                      Modified = cbModified
                Catch uhException As Exception
                      Throw
                End Try
```

```
            End Get
        End Property

End Class
```

A generated business class - CBApplication

```
Option Explicit On
Option Strict On

Public Class CBApplication
      Inherits CDApplication

      Public Sub New(psUser As String)
                  MyBase.New(psUser)
      End Sub

      Public Sub New(ByVal psUser As String, ByVal pbiApplicationId AS
Int64)
            MyBase.New(psUser, pbiApplicationId)
      End Sub

      Public Sub New(ByVal psUser As String, ByVal pbiPositionId AS Int64,
ByVal psLastName AS String, ByVal psFirstName AS String, ByVal
psPrimaryPhone AS String)
            MyBase.New(psUser, pbiPositionId, psLastName, psFirstName,
psPrimaryPhone)
      End Sub

      Public ReadOnly Property Position As CBPosition
            Get
                  Try
                        Position = New CBPosition("", Me.PositionId)
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
      End Property

      Public ReadOnly Property ApplicantStatus As CBApplicantStatus
            Get
                  Try
                        ApplicantStatus = New CBApplicantStatus("",
Me.ApplicantStatusId)
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
      End Property

      Public Overloads Property PositionId(psUser As String) As Int64
            Get
                  Try
                        PositionId = MyBase.PositionId
                  Catch uhException As Exception
                        Throw
                  End Try
            End Get
            Set(ByVal Value As Int64)
```

```
                Dim lobjPosition As CBPosition
                Try
                      lobjPosition = New CBPosition(psUser, Value)
                      MyBase.PositionId = Value
                Catch appException As ApplicationException
                      Throw New ApplicationException("Invalid
PositionId.")
                Catch uhException As Exception
                      Throw
                End Try
          End Set
    End Property

    Public Overloads Property ApplicantStatusId(psUser As String) As
Int64
          Get
                Try
                      ApplicantStatusId = MyBase.ApplicantStatusId
                Catch uhException As Exception
                      Throw
                End Try
          End Get
          Set(ByVal Value As Int64)
                Dim lobjApplicantStatus As CBApplicantStatus
                Try
                      lobjApplicantStatus = New CBApplicantStatus(psUser,
Value)
                      MyBase.ApplicantStatusId = Value
                Catch appException As ApplicationException
                      Throw New ApplicationException("Invalid
ApplicantStatusId.")
                Catch uhException As Exception
                      Throw
                End Try
          End Set
    End Property

  Public Overloads Sub Save(ByVal pobjApplication As CBApplication)
      MyBase.Save()
  End Sub

End Class
```

A custom class - PageRole

```
Public Class PageRole
    Public Function IsValidPageRole(ByVal psURL As String) As Boolean
        Dim lsPageName As String
        Dim lbiRoleId As Int64
        Dim lobjRole As New CBRole("")

        'Ensure that the employee's current role is valid for the desired
page

        'Get the page name
        If psURL.LastIndexOf("/") > 0 Then
            lsPageName = psURL.Substring(psURL.LastIndexOf("/") + 1)
        Else
            lsPageName = psURL
        End If

        If lsPageName.IndexOf("?") > 0 Then
            lsPageName = lsPageName.Remove(lsPageName.IndexOf("?"))
        End If

        ' Get the role information
        lbiRoleId = Convert.ToInt64(SessionVars.RoleId)

        If lbiRoleId = lobjRole.GetRecruitingCoordinator Then ' Recruiting
Coordinator can do everything
            Return True
        End If
        Select Case lsPageName.ToLower()
            Case "home.aspx"
                Return True ' everyone can view the home page
            Case "openposition.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "editposition.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "closeposition.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "listpositions.aspx"
                Return True ' everyone can list positions
            Case "createcandidate.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "editcandidate.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "listcandidates.aspx"
                Return True ' everyone can list candidates
            Case "selectpositionlistapplicants.aspx"
                Return True ' everyone can list candidates
            Case "applicationconfirmation.aspx"
                Return True ' everone can list candidates
            Case "displaytextfile.aspx"
                Return True ' everyone can list candidates
            Case "addsubscription.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager)
```

```
            Case "removesubscription.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager)
            Case "requestinitialinterview.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager)
            Case "subscriptionconfirmation.aspx"
                Return True
            Case "updatecandidatestatus.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager)
            Case "selectpositionforcandidateupdate.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager)
            Case "selectapplicantforupdate.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager)
            Case "listsubscriptions.aspx"
                Return True
            Case "schedulecandidateinterview.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "selectapplicantforinterview.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "selectpositionforcandidateinterview.aspx"
                Return (lbiRoleId = lobjRole.GetRecruitingCoordinator())
            Case "schedulecandidateinterviewconfirm.aspx"
                Return True
            Case "schedulecandidatefollowup.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager())
            Case "schedulecandidatefollowupconfirm.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager())
            Case "selectpositionforcandidatefollowup.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager())
            Case "selectapplicantforfollowup.aspx"
                Return (lbiRoleId = lobjRole.GetHiringManager())
            Case ""
                Return True
            Case Else
                Throw New System.ApplicationException("Unknown page: " +
lsPageName)
        End Select
    End Function

End Class
```

APPENDIX C: UI CODE EXAMPLES

The ASP.Net Master page

```
<%@ Master Language="VB" AutoEventWireup="false"
CodeBehind="DP08.Master.vb" Inherits="DP08CandidateTracking_Intranet.DP08"
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>CTS</title>
    <link rel="stylesheet" type="text/css" href="DP08.css" />
</head>
<body>
<form id="masterform" runat="server">
    <div class="header">
        <div style="width:100%; text-align:center">
          <b>Candidate Tracking System</b>
        </div>
        <div style="float: left; width:49%; text-align:left;">
          <b>Welcome:</b> <asp:Label ID="lblUserName" runat="server"
Text=""></asp:Label>
        </div>
        <div style="float: left; width:49%; text-align:right;">
          <b>Agency:</b> <asp:Label ID="lblAgency" runat="server"
Text=""></asp:Label> 
          <b>Division:</b> <asp:Label ID="lblDivision" runat="server"
Text=""></asp:Label> 
          <b>Role: </b><asp:Label ID="lblRole" runat="server"
Text=""></asp:Label>
        </div>
    </div>
    <div class="navigation">
        <asp:SiteMapDataSource ID="SiteMapDataSourceMain" runat="server"
            ShowStartingNode="False" />
        <asp:TreeView ID="TreeViewNav" runat="server"
DataSourceID="SiteMapDataSourceMain"
            ImageSet="Simple" NodeIndent="10" BorderWidth="0px"
ExpandDepth="0"
            NodeWrap="True" PopulateNodesFromClient="False">
            <ParentNodeStyle Font-Bold="False" />
            <HoverNodeStyle Font-Underline="True" ForeColor="#DD5555" />
            <SelectedNodeStyle Font-Underline="True" ForeColor="#DD5555"
            HorizontalPadding="0px" VerticalPadding="0px" />
            <NodeStyle Font-Names="Verdana" Font-Size="8pt"
ForeColor="black"
            HorizontalPadding="0px" NodeSpacing="0px" VerticalPadding="0px"
/>
        </asp:TreeView>
```

```
    </div>

    <div class="main">
        <asp:ContentPlaceHolder id="cntPlaceHolderBody" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    <div class="footer">
      <br />
      <br />
    </div>
</form>
</body>
</html>
```

The ASP.Net Master page code-behind .vb file

```
Partial Public Class DP08
    Inherits System.Web.UI.MasterPage
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim lobjPageRole As New DP08CandidateTracking_ClassLibrary.PageRole

        If Not lobjPageRole.IsValidPageRole(Request.Url.ToString) Then
            Response.Redirect("Unauthorized.aspx", True)
        End If
        ' Set the header info on the page
        lblUserName.Text =
DP08CandidateTracking_ClassLibrary.SessionVars.EmployeeName
        lblAgency.Text =
DP08CandidateTracking_ClassLibrary.SessionVars.AgencyAbbreviation
        lblDivision.Text =
DP08CandidateTracking_ClassLibrary.SessionVars.DivisionName
        lblRole.Text =
DP08CandidateTracking_ClassLibrary.SessionVars.RoleName
    End Sub

    Protected Sub TreeViewNav_TreeNodeDataBound(ByVal sender As Object,
ByVal e As System.Web.UI.WebControls.TreeNodeEventArgs) Handles
TreeViewNav.TreeNodeDataBound
        Dim url As String = e.Node.NavigateUrl

        If url <> "" Then
            If url.StartsWith(Request.ApplicationPath) Then
                url = url.Replace(Request.ApplicationPath, "~")
            Else
                If Not url.StartsWith("~/") Then
                    url = "~/" + url
                End If
            End If
            e.Node.NavigateUrl = Me.ResolveUrl(url)
        End If
        'Check to see if the SiteMap Node needs to be NOT visible.
        Dim tree As System.Web.UI.WebControls.TreeView = DirectCast(sender,
System.Web.UI.WebControls.TreeView)
        Dim sNode As SiteMapNode = DirectCast(e.Node.DataItem, SiteMapNode)
        Dim lobjPageRole As New DP08CandidateTracking_ClassLibrary.PageRole

        If Not lobjPageRole.IsValidPageRole(sNode.Url.ToString) Then
            Dim parent As System.Web.UI.WebControls.TreeNode =
e.Node.Parent
            If parent IsNot Nothing Then
                'Remove the Node from the TreeView.
                parent.ChildNodes.Remove(e.Node)
            End If
        End If
    End Sub

End Class
```

A typical data entry .aspx page – EditCandidate.aspx

```
<%@ Page Title="" Language="vb" AutoEventWireup="false"
MasterPageFile="~/DP08.Master" CodeBehind="EditCandidate.aspx.vb"
Inherits="DP08CandidateTracking_Intranet.EditCandidate" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cntPlaceHolderBody"
runat="server">
<div style="width:100%; margin: 5px">
  <div style=" width:100%;margin: 5px;">
    <div class="pagetitle">
      Edit Candidate
    </div>
    <div id="errorarea" style="width:100%;display:none; margin:5px;
color:Red" runat="server">
      <asp:Label ID="lblErrorMsg" runat="server" Text=""></asp:Label>
    </div>
    <div id="workarea" style="width:100%;margin: 5px;">
      <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Position Type:
          </div>
          <div class="horizontalinput">
            <asp:Label ID="lblPositionType" runat="server"
Text=""></asp:Label>
          </div>
      </div>
      <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Open Date:
          </div>
          <div class="horizontalinput">
            <asp:Label ID="lblOpenDate" runat="server" Text="">
</asp:Label>
          </div>
      </div>
      <div class="horizontalcontrolgroup">
        <div class="horizontallabel">
          Date of Interest:
        </div>
        <div class="horizontalinput">
            <asp:Calendar ID="calDateOfInterest" runat="server" TitleStyle-
BackColor="#87CEFA" SelectedDayStyle-BackColor="#87CEFA"></asp:Calendar>
        </div>
      </div>
      <div class="horizontalcontrolgroup">
        <div class="horizontallabel">
          First Name:
        </div>
        <div class="horizontalinput">
          <asp:TextBox ID="txtFirstName" MaxLength="50" Columns="30"
runat="server"></asp:TextBox>
          <asp:RegularExpressionValidator ID="regexvalFirstName"
               ControlToValidate="txtFirstName"
               ValidationExpression="^[a-zA-Z''-'\s]{1,20}$"
```

```
                ErrorMessage="A name may ony contain a-z, A-Z, single quote
and hyphen characters."
                EnableClientScript="true" runat="server" />
          </div>
        </div>
        <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Last Name:
          </div>
          <div class="horizontalinput">
            <asp:TextBox ID="txtLastName" MaxLength="50" Columns="30"
runat="server"></asp:TextBox>
            <asp:RegularExpressionValidator ID="regexvalLastName"
                ControlToValidate="txtLastName"
                ValidationExpression="^[a-zA-Z''-'\s]{1,20}$"
                ErrorMessage="A name may ony contain a-z, A-Z, single quote
and hyphen characters."
                EnableClientScript="true" runat="server" />
          </div>
        </div>
        <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Primary Phone:
          </div>
          <div class="horizontalinput">
            <asp:TextBox ID="txtPrimaryPhone" MaxLength="20" Columns="20"
runat="server"></asp:TextBox>
            Format: 605.773.1234 x5678
            <asp:RegularExpressionValidator ID="regexpvalPrimaryPhone"
                ControlToValidate="txtPrimaryPhone"
                ValidationExpression="^\d{3}\.\d{3}\.\d{4}(\s[xX]?[0-
9]{1,4})?$"
                ErrorMessage="Primary Phone must be in 605.773.1234 format
with optional extension in x5678 format."
                EnableClientScript="true" runat="server" />
          </div>
        </div>
        <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Alternate Phone:
          </div>
          <div class="horizontalinput">
            <asp:TextBox ID="txtAlternatePhone" MaxLength="20" Columns="20"
runat="server"></asp:TextBox>
            Format: 605.773.1234 x5678
            <asp:RegularExpressionValidator ID="regexvalAlternatePhone"
                ControlToValidate="txtAlternatePhone"
                ValidationExpression="^\d{3}\.\d{3}\.\d{4}(\s[xX]?[0-
9]{1,4})?$"
                ErrorMessage="Alternate Phone must be in 605.773.1234 format
with optional extension in x5678 format."
                EnableClientScript="true" runat="server" />
          </div>
        </div>
        <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Email Address:
```

```
        </div>
        <div class="horizontalinput">
          <asp:TextBox ID="txtEmailAddress" MaxLength="50" Columns="30"
runat="server"></asp:TextBox>
          <asp:RegularExpressionValidator ID="regexvalEmailAddress"
               ControlToValidate="txtEmailAddress"
               ValidationExpression="^([0-9a-zA-Z]([-\.\w]*[0-9a-zA-
Z])*@([0-9a-zA-Z][-\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$"
               ErrorMessage="Invalid email format."
               EnableClientScript="true" runat="server" />
        </div>
      </div>
      <div class="horizontalcontrolgroup">
          <div class="horizontallabel">
            Candidate Status:
          </div>
          <div class="horizontalinput">
            <asp:DropDownList ID="ddlApplicantStatus" runat="server">
            </asp:DropDownList>
          </div>
      </div>
      <div class="horizontalcontrolgroup">
        <div class="horizontallabel">
          Application Date:
        </div>
        <div class="horizontalinput">
          <asp:Calendar ID="calApplicationDate" runat="server" TitleStyle-
BackColor="#87CEFA" SelectedDayStyle-BackColor="#87CEFA"></asp:Calendar>
        </div>
      </div>
      <div class="horizontalcontrolgroup">
        <div class="horizontallabel">
          Resume:
        </div>
        <div class="horizontalinput">
          <asp:FileUpload ID="fupldResume" runat="server" />
        </div>
      </div>
      <div class="horizontalcontrolgroup">
        <div class="horizontallabel">
          Application:
        </div>
        <div class="horizontalinput">
          <asp:FileUpload ID="fupldApplication" runat="server" />
        </div>
      </div>
      <div class="horizontalcontrolgroup">
        <div class="horizontallabel">
          Cover Letter:
        </div>
        <div class="horizontalinput">
            <asp:FileUpload ID="fupldCoverLetter" runat="server" />
        </div>
      </div>
      <div class="horizontalcontrolgroup">
        <asp:Button class="button" ID="btnSaveChanges" runat="server"
Text="Save Changes" />
```

```
        <asp:Button class="button" ID="btnCancelChanges"
CausesValidation="false" runat="server" Text="Cancel Changes" />
      </div>
    </div>
  </div>
</div>
</asp:Content>
```

A typical code-behind .vb file – EditCandidate.aspx.vb

```
Imports DP08CandidateTracking_ClassLibrary
Partial Public Class EditCandidate
    Inherits BasePage

    Protected Overloads Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If Not IsPostBack Then
            Dim lbiApplicationId As Int64
            Dim lobjApplication As CBApplication

            'check the query string
            lbiApplicationId =
Convert.ToInt64(Request.QueryString("ApplicationId"))
            If lbiApplicationId < 1 Then
                Response.Redirect("SelectApplicantForEdit.aspx", True)
            End If

            ' Set the session with the ApplicationId
            SessionVars.ApplicationId = lbiApplicationId

            lobjApplication = New CBApplication("", lbiApplicationId)
            ' Set the sesion with the PositionId
            SessionVars.PositionId = lobjApplication.PositionId

            ' Get the existing position information
            Dim lobjPosition As New CBPosition("",
lobjApplication.PositionId)
            lblPositionType.Text = lobjPosition.PositionType.Name
            lblOpenDate.Text = lobjPosition.OpenDate.ToShortDateString
            ' Initialize the controls
            calDateOfInterest.SelectedDate =
lobjApplication.DateOfInterest.Date

LoadApplicantStatus(lobjApplication.ApplicantStatus.Name.ToString)
            ' Setup the applicant files (resume, application, cover letter)
            Me.txtFirstName.Text = lobjApplication.FirstName
            Me.txtLastName.Text = lobjApplication.LastName
            Me.txtPrimaryPhone.Text = lobjApplication.PrimaryPhone
            Me.txtEmailAddress.Text = lobjApplication.EmailAddress
            Me.txtAlternatePhone.Text = lobjApplication.AlternatePhone
        End If

    End Sub
    Private Sub LoadApplicantStatus(ByVal pApplicantStatus As String)
        Dim dsApplicantStatus As DataSet
        Dim dvApplicantStatus As DataView
        Dim drvApplicantStatus As DataRowView
        Dim lcolApplicantStatus As New CCApplicantStatus
        Dim i As Integer = 0

        Try
```

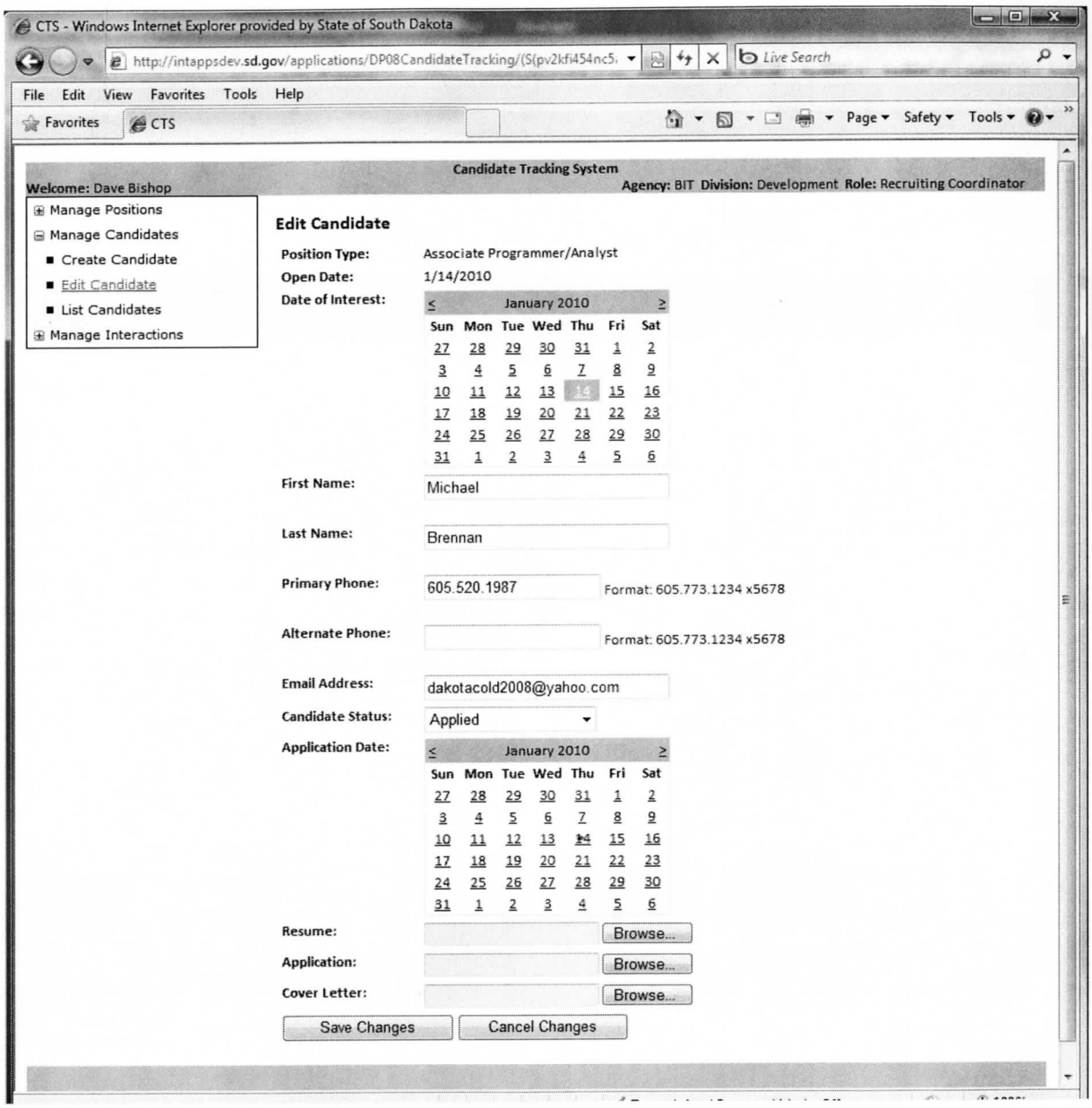
```
            dsApplicantStatus =
lcolApplicantStatus.GetActiveApplicantStatus()
            dvApplicantStatus = New DataView(dsApplicantStatus.Tables(0))
            For Each drvApplicantStatus In dvApplicantStatus
                Me.ddlApplicantStatus.Items.Add(New
ListItem(drvApplicantStatus("Name").ToString,
drvApplicantStatus("ApplicantStatusId").ToString))
                If drvApplicantStatus("Name").ToString.ToUpper =
pApplicantStatus.ToUpper Then
                    Me.ddlApplicantStatus.Items(i).Selected = True
                End If
                i = i + 1
            Next
        Catch ex As Exception
        Finally
            lcolApplicantStatus = Nothing
            dvApplicantStatus = Nothing
        End Try
    End Sub
    Private Sub btnSaveChanges_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSaveChanges.Click
        Dim lobjApplication As CBApplication
        Dim sFileExtension As String
        Dim sFileName As String

        ' Reset the error message area
        lblErrorMsg.Text = ""
        Me.errorarea.Style("display") = "none"

        Try
            lobjApplication = New CBApplication("",
Convert.ToInt64(SessionVars.ApplicationId))
            lobjApplication.PositionId = lobjApplication.PositionId
            lobjApplication.DateOfInterest = calDateOfInterest.SelectedDate
            lobjApplication.ApplicantStatusId =
ddlApplicantStatus.Items(ddlApplicantStatus.SelectedIndex).Value
            lobjApplication.FirstName = txtFirstName.Text
            lobjApplication.LastName = txtLastName.Text
            lobjApplication.PrimaryPhone = txtPrimaryPhone.Text
            lobjApplication.AlternatePhone = txtAlternatePhone.Text
            lobjApplication.EmailAddress = txtEmailAddress.Text
            lobjApplication.AppReceivedDate =
calApplicationDate.SelectedDate

            If fupldResume.HasFile Then
                sFileExtension = GetFileExtension(fupldResume.FileName)
                sFileName = lobjApplication.ApplicationId.ToString +
"Resume." + sFileExtension
                fupldResume.SaveAs(Server.MapPath("~/ApplicantFiles/") +
sFileName)
                lobjApplication.ResumeFile = sFileName
            End If
            If fupldApplication.HasFile Then
                sFileExtension =
GetFileExtension(fupldApplication.FileName)
                sFileName = lobjApplication.ApplicationId.ToString +
"Application." + sFileExtension
```

```
                fupldApplication.SaveAs(Server.MapPath("~/ApplicantFiles/")
+ sFileName)
                lobjApplication.ApplicationFile = sFileName
            End If
            If fupldCoverLetter.HasFile Then
                sFileExtension =
GetFileExtension(fupldCoverLetter.FileName)
                sFileName = lobjApplication.ApplicationId.ToString +
"CoverLetter." + sFileExtension
                fupldCoverLetter.SaveAs(Server.MapPath("~/ApplicantFiles/")
+ sFileName)
                lobjApplication.CoverLetterFile = sFileName
            End If

            lobjApplication.Save(lobjApplication)
            Response.Redirect("ApplicationConfirmation.aspx", True)

        Catch AppEx As ApplicationException
            If AppEx.Message = "PositionTypeId is required." Then
                lblErrorMsg.Text = "A Position Type must be selected."
                Me.errorarea.Style("display") = "block"
            Else
                lblErrorMsg.Text = AppEx.Message
                Me.errorarea.Style("display") = "block"
            End If

        Catch SQLEx As System.Data.SqlClient.SqlException
            Select Case SQLEx.ErrorCode
                Case -2146232060 ' Duplicate row
                    lblErrorMsg.Text = "This candidate already exists."
                    Me.errorarea.Style("display") = "block"
                Case Else
                    lblErrorMsg.Text = SQLEx.ErrorCode + " " +
SQLEx.Message
                    Me.errorarea.Style("display") = "block"
            End Select
        Catch ex As Exception
            lblErrorMsg.Text = ex.ToString
            Me.errorarea.Style("display") = "block"
        Finally
            lobjApplication = Nothing
        End Try

    End Sub

    Private Sub btnCancelChanges_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnCancelChanges.Click
        SessionVars.ApplicationId = 0
        Response.Redirect("ApplicationConfirmation.aspx", True)
    End Sub
End Class
```

APPENDIX B: USER INTERFACE IMAGES

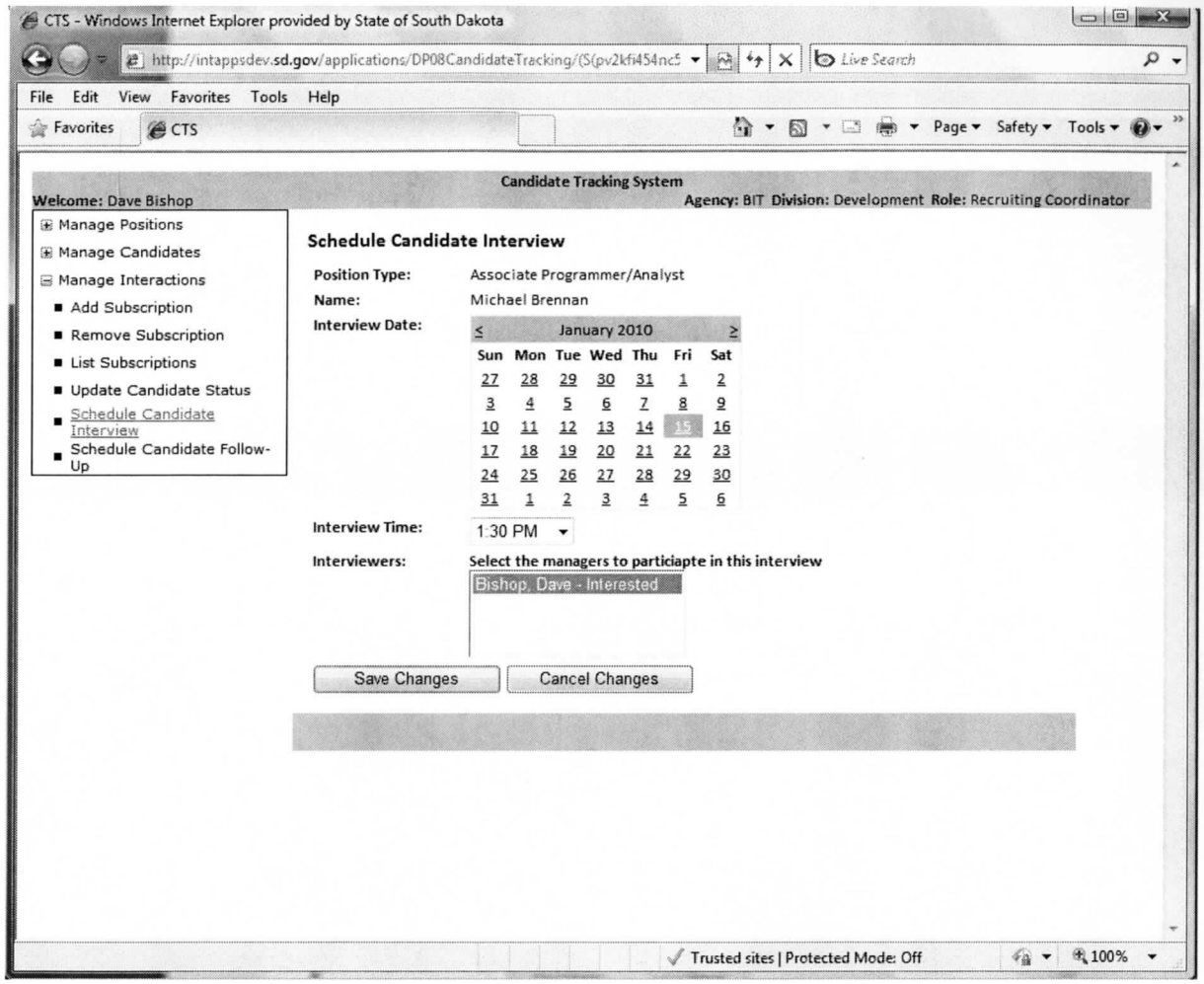
The Edit Candidate page

Figure 7 - Edit Candidate Screenshot



The Schedule Candidate Interview page

Figure 8 - Schedule Candidate Interview Screenshot



The Update Candidate Status page – Recording Interview Results

Figure 9 - Update Candidate Status Screenshot